Hybrid Optical Inference Machines

Final Report
AFOSR-86-0301

27 September 1991

Cardinal Warde

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA

Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203

and

Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D.C. 20332

Approved for Public Release
Distribution Unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
UNLIMITED

. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

a. NAME OF PERFORMING ORGANIZATION
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

6c. ADDRESS *(City, State, and ZIP Code)*
77 MASSACHUSETTS AVE.
CAMBRIDGE, MA 02139

7b. ADDRESS *(City, State, and ZIP Code)*
BOLLING AFB DC 20332
Bldg 410

a. NAME OF FUNDING / SPONSORING ORGANIZATION
DARPA

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR - 86 - 0301

c. ADDRESS *(City, State, and ZIP Code)*
3701 N. FAIRFAX DRIVE
ARLINGTON, VA 2203

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61101E	5780	03	

1. TITLE *(Include Security Classification)*
HYBRID OPTICAL INFERENCE MACHINES (UNCLASSIFIED)

2. PERSONAL AUTHOR(S)
PROF. CARDINAL WARDE, DR. JAMES KOTTAS, MR. VERNON SHRAUGER

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM 8/15/86 TO 2/14/91

14. DATE OF REPORT *(Year, Month, Day)*
1991,09,27

15. PAGE COUNT
92

6. SUPPLEMENTARY NOTATION

7. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Inference engines, neural networks, back propagation, limit cycles, optical interconnections, interconnection holograms.

.9. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

- See next page --

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

2a. NAME OF RESPONSIBLE INDIVIDUAL
~~PROF. CARDINAL WARDE~~ Giles

22b. TELEPHONE *(Include Area Code)*
~~617/253-6858~~ 202-767-4931

22c. OFFICE SYMBOL
NE

Previous editions are obsolete.

Abstract

This program has investigated the use of limit cycles to represent and processing symbolic information in the context of an inference machine. This approach was proposed as a means of overcoming problems with fault tolerance and relatively small space-bandwidth products in current spatial light modulator (SLM) technology. The program has focused on developing a storage medium with many limit cycles (oscillatory modes) available and a method for coupling the various modes in a desired way. Because of their flexibility, neural network ideas were used as the basis for the components and algorithms developed.

In the theoretical realm, the program has had many accomplishments. First, the self-oscillating neural network (SONN) model was developed and characterized as the oscillatory medium. This model was designed with optical spatial SLMs in mind and does not require any training or programming. Furthermore, it is highly tolerant of static parameter variations inherent in the optics.

Next, the spectral back-propagation (SBP) training algorithm was developed with complete generality as a means of forming the coupling trajectories. This algorithm trains input-output sequences into a network using an error criterion based on a Fourier series decomposition of the sequences. The method allows the interconnects to have trainable time delays in addition to the weights. This capability proved very beneficial when developing a transition-detecting network for realizing the mode couplings. The algorithm also allows the cells to have finite bandwidth.

Both the SONN and the SBP algorithm were combined to demonstrate a simple symbolic processing system based on limit cycles. The chosen paradigm was a finite state machine (FSM), a simple starting point for building up to a complete inference machine.

With respect to optics, an optical architecture for the SONN model was designed. This architecture is effectively a specialized optical neural network based on holographic interconnects between SLMs. To satisify architectural demands, the program has developed a method for generating interconnection holograms using a computer and current color printer technology. The holograms are phase-only, have very high efficiency, require low cost processing facilities, and are expediently made.

Contents

1 Introduction

The goal of this research program was to investigate the use of optics in symbolic processing systems, and in particular, inference machines. These systems store information in the form of relationships between symbols, usually arranged as a knowledge base of rules. Their function is to infer the answers to queries of the knowledge base by searching through the set of relationships. The structure and function of an inference machine and the necessary considerations for an optical implementation are discussed in more detail in Section 8.

The operational requirements for an inference machine are (1) to store many rules (i.e., have a large knowledge base capacity) and (2) to search the knowledge base very quickly. Unlike numerical processors such as matrix-vector multipliers, inference machines do not require large dynamic range. Thus, the parallelism and speed of optics offered an attractive implementation technology.

The initial optical architectures for an optical inference machine were based on thresholding matrix-vector multipliers. These designs are described in more detail in Section 9. Unfortunately, these architectures suffered from two major limitations: small capacity and low fault tolerance. A rule was represented by a matrix that encoded a particular relationship between vectors of symbols. In the optical implementation, the rule matrices were stored on spatial light modulators (SLMs). Therefore, the size of the rule matrices and thus the symbol vectors was limited by the size of current SLM technology. Furthermore, if a pixel on the SLM failed, the corresponding element in all the rule matrices would be altered permanently, thus causing all rules to be changed in an undesired way.

We proposed to consider a very different approach to designing an inference machine for an optical implementation in hopes of solving these two problems. Instead of representing symbols using fault-*in*tolerant vectors (fixed points), we have investigated the use of limit cycles for this role. This form constitutes a dynamic, temporal representation of information. A known disadvantage of this method is reduced access time since multiple points along a trajectory must be observed in order to recognize the current cycle (if one is even active).

In order to construct an inference machine based on limit cycles, two fundamental components are needed. The first one is a medium with many limit cycles (i.e., oscillatory modes) available. Each cycle would represent a different symbol or a logical state of the machine. Ideally, this component would not require any programming or training. We successfully developed the *self-oscillating neural network* (SONN) model for this purpose. This model is designed with an optical SLM implementation in mind. It is described in Section 2.

The second component is a method for coupling the cycles. Ideally, this method (1) is not dependent upon cycle shapes, (2) can be reprogrammed or retrained as needed or desired, and (3) does not perform any explicit conversions from a limit cycle (LC) to a fixed point (FP).[1] We developed the *spectral back-propagation* (SBP) training algorithm and a transition-detecting network for this purpose. Functionally, a transition detector

[1] This requirement is not necessary for a practical machine. It is included here to focus on the study of using oscillatory phenomena to represent and process information.

achieves the first two characteristics, but not the third one. The SBP algorithm is described with complete generality in Section 3

We demonstrated these concepts by simulating a finite state machine (FSM) based on limit cycles (an LC-FSM). This computation paradigm offers a simple starting point for building up to a complete inference machine. The simulated LC-FSM used the SONN both as an associative memory for limit cycles and as an input source for the LC-FSM. In addition, the LC-FSM illustrated SBP-trained transition detectors for recognizing transition conditions. This work is discussed in Section 4.

The optical implementation considerations have focused on the SONN model because of the latter's robustness. An optical architecture for the SONN has been designed and is discussed in Section 5. Besides the SLMs, the key component of this architecture are the interconnect holograms. Practical constraints require high efficiency interconnect holograms which motivated the development of a new approach to computer generating interconnection holograms. This process we have developed takes advantage of current color printer technology as a mechanism for modulating the exposure of black and white film. Computer generated color masks representing desired phase-only functions were photoreduced onto high resolution black and white, and after developing and bleach processing, each of the printer colors map to discrete phase levels as required by the specified phase-only function. A total of 8 colors yielded a total of 8 discrete phase levels which were used to construct arbitrary synthetic blazed grating interconnections exhibiting efficiencies of atleast 50%. In addition, a computer-controlled system for characterizing SLM parameters was developed. This work also is discussed in Section 5.

2 Self-Oscillating Neural Network Model

The SONN model is an oscillatory medium with many modes (i.e., limit cycles) available naturally. No training or programming is required. Furthermore, the existence of the modes is highly tolerant of static parameter variations in the network parameters.

The structure of the SONN in shown in Figure 1. The network is a hierarchical arrangement of smaller feedforward networks called levels. Connections going up the hierarchy are excitatory with value $C_L = \frac{1}{2}$. Similarly, connections going down the hierarchy are inhibitory with value $-H_L = -1$.

Each level uses the off-center, on-surround canonical interconnect topology shown in Figure 2. The central inhibitory interconnect has a weight $-H_l = -1$ and the off-center excitatory interconnects have weights $C_l = \frac{1}{6}$. For the levels shown in Figure 1, these values correspond to a ratio of the total excitation to the total inhibition of $\frac{1}{2}$. Simulations showed that this ratio value is a good choice for robust oscillatory behavior to exist.

The cells in the SONN follow the model shown in Figure 3. They are governed by the discrete-time equations,

$$u_i[t] = \sum_j w_{ij} y_j[t-T_{ij}-1] + x_i[t], \qquad (1)$$

$$v_i[t] = \alpha_v v_i[t-1] + (1-\alpha_v) u_i[t], \qquad (2)$$

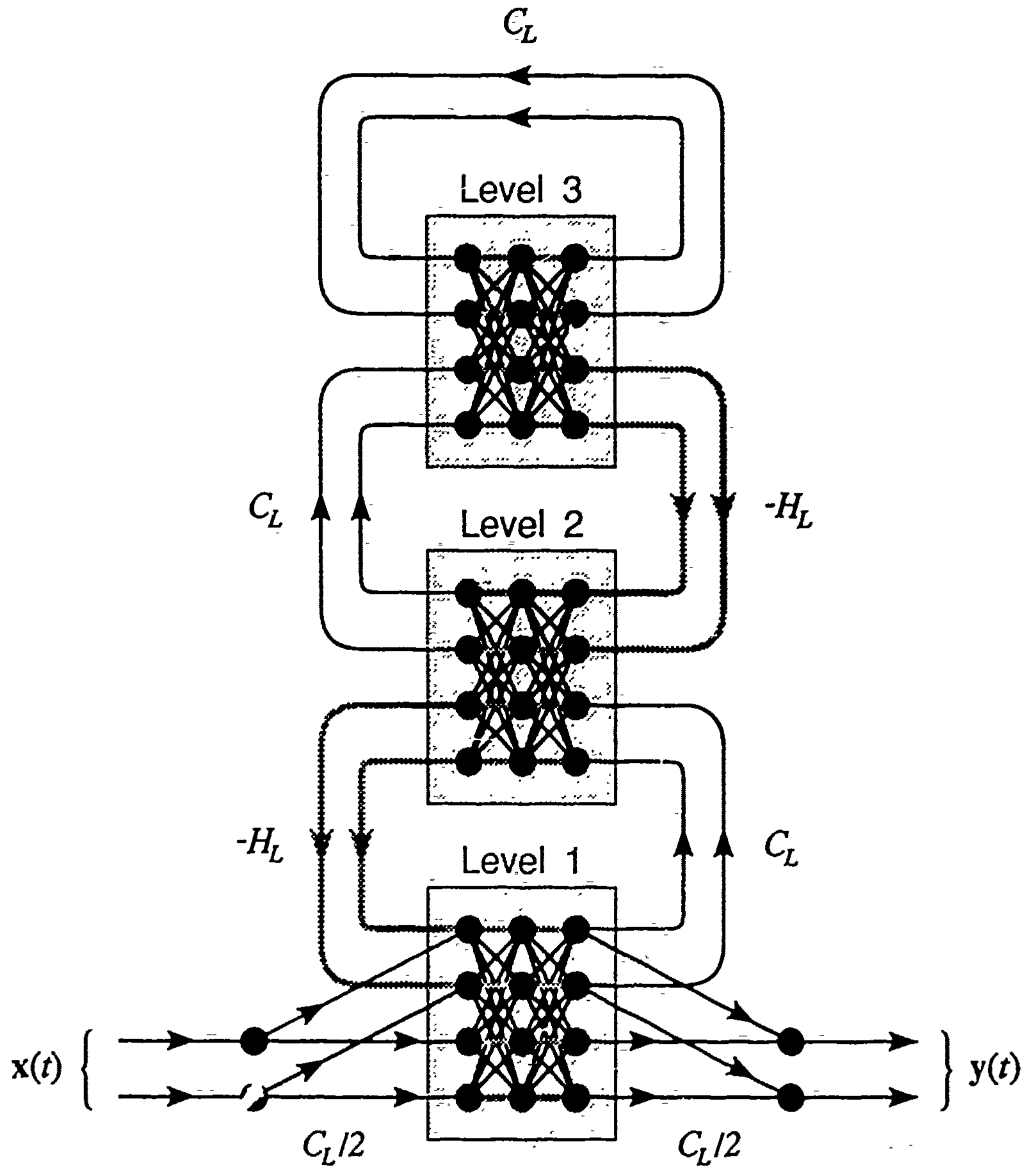


Figure 1: Self-oscillating neural network (SONN). Solid lines indicate excitatory interconnects whereas shaded lines denote inhibitory interconnects.

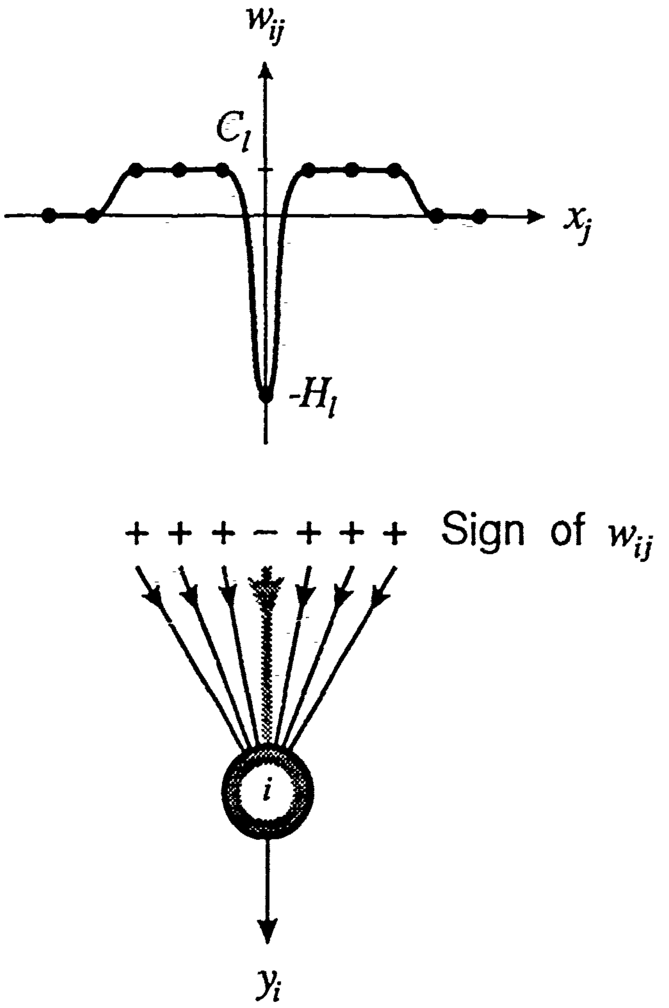


Figure 2: Off-center, on-surround canonical interconnect topology used in each level.

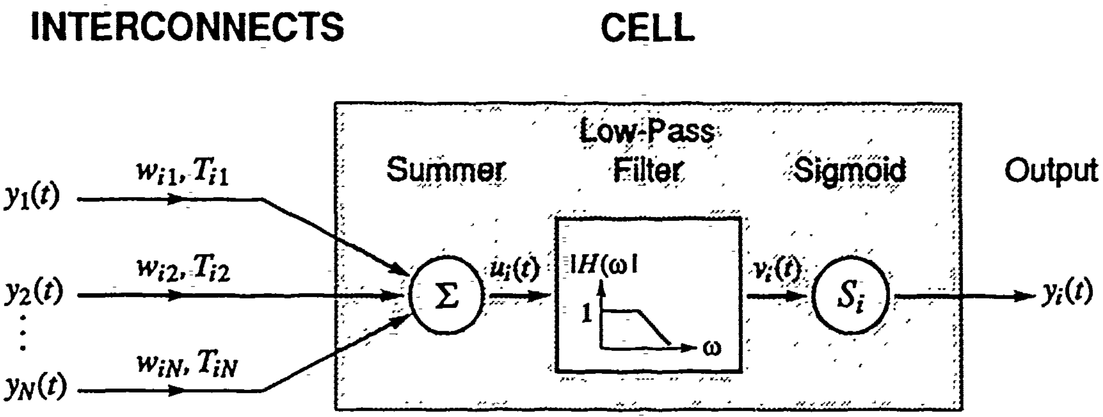


Figure 3: Cell model used in the SONN.

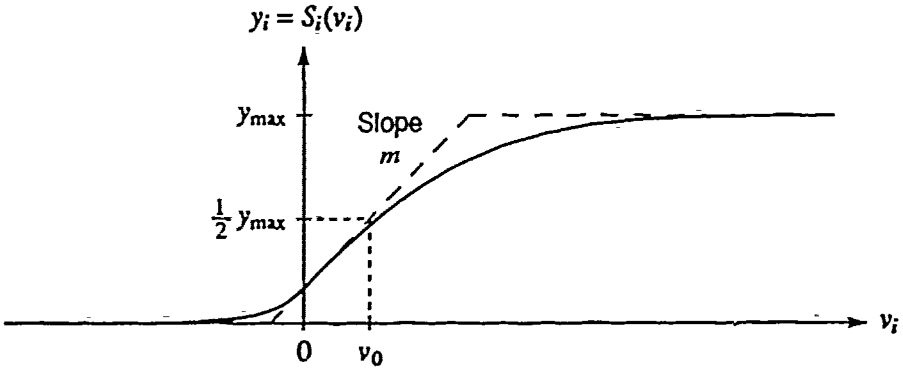


Figure 4: Nominal sigmoidal cell output function S_i.

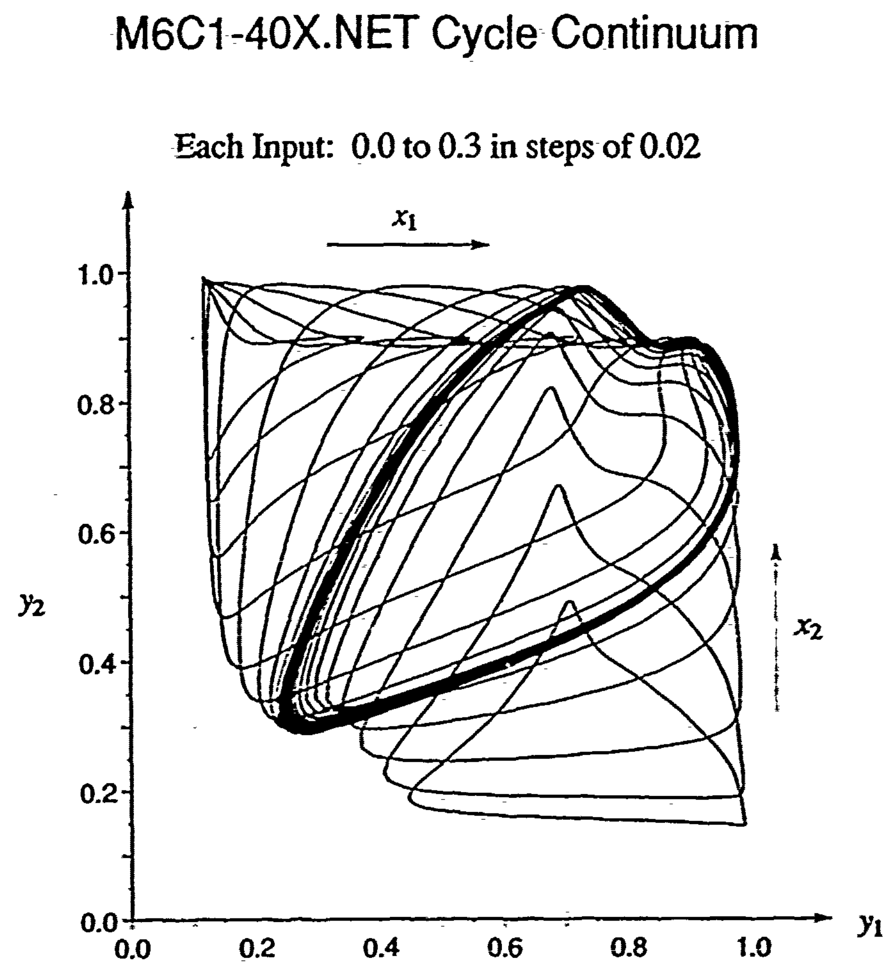


Figure 5: Sampling of the continuum of cycles available in the SONN with constant inputs. The weights were perturbed randomly by ±20% from the nominal values. Similarly, the time delays were randomly selected from {0,1,2,3,4} iterations.

$$y_i[t] \;=\; S_i\big(v_i[t]\big) \tag{3}$$

where $u_i[t]$ is the weighted input sum for the i^{th} cell, $v_i[t]$ is the filtered input sum with damping factor α_v, $y_i[t]$ is the cell output as formed by the output function S_i, w_{ij} is the weight associated with the interconnect from the j^{th} cell, and T_{ij} is the time delay associated with the same interconnect. The nominal shape for a sigmoidal output function is shown in Figure 1. For simplicity, the time delays, if present, are taken to be integers here. In the simulations, the damping factor α_v was set to 0.7 which corresponded to an effective time constant of approximately 2.8 iterations.

A mode can be selected with either a constant (i.e., FP) or cyclical (i.e., LC) input. Using constant inputs, a sampling of the cycles available is shown in Figure 5. The particular SONN that generated these cycles had its weights randomly perturbed by up to ±20% from their nominal values. Furthermore, the network had a distribution of time delays throughout the network with T_{ij} ranging from 0 to 4 iterations with a mean of 2 iterations. The periods of the cycles generated by the SONN used in Figure 5 were all approximately 64 iterations.

Simulations showed that (1) variations in the interconnect time delays create diverse cycle shapes and (2) the SONN easily can tolerate static variations of over ±20% in its network parameters (weights, sigmoid maxima and gains, etc.).

The optical implementation of the SONN is discussed in Section 5.

3 Spectral Back-Propagation Training Algorithm

3.1 Overview

The SBP training algorithm is an extension of the conventional back-propagation method [1] for training a neural network to learn a set of smooth input-output sequences. It can adapt both the weights and the time delays or any combination thereof. It can work with either feedforward or recurrent networks. In addition, the cells can have infinite or finite bandwidth using a first order approximation as in Equation (2).

The algorithm has been demonstrated successfully using computer simulations for several different cases. It has trained a simple recurrent network with an infinite impulse response (IIR) to learn continuous-time cycles. Similarly, it can train either the weights or the time delays of a finite impulse response (FIR) network. Finally, it can train a conventional feedforward network with vector input and output patterns.

3.2 Mathematical Basis for the Algorithm

Functionally, the SBP algorithm compares the spectral decomposition of the actual output sequences to that of the desired output sequences to form an error measruement for driving the adaptation.

Consider the continuous-time form of the cell equations given in (1)–(3):

$$u_i(t) = \sum_j w_{ij} y_j(t-T_{ij}) + x_i(t), \tag{4}$$

$$\tau_v \frac{dv_i(t)}{dt} = -v_i(t) + u_i(t), \tag{5}$$

$$y_i(t) = S_i\big(v_i(t)\big), \tag{6}$$

where $\alpha_v = e^{-1/\tau_v}$. Assuming the output $y_i(t)$ is smooth and "slowly varying," it can be approximated by the truncated Fourier series,

$$y_i(t) \approx \sum_{k=0}^{K} \left[Y_{ik}^c \cos(k\omega_0 t) + Y_{ik}^s \sin(k\omega_0 t)\right], \tag{7}$$

where

$$Y_{ik}^c = \frac{\beta_k}{T_0} \int_0^{T_0} y_i(t) \cos(k\omega_0 t)\, dt, \tag{8}$$

$$Y^s_{ik} = \frac{\beta_k}{T_0}\int_0^{T_0} y_i(t)\sin(k\omega_0 t)\,dt, \tag{9}$$

$$\beta_k = \begin{cases} 1 & \text{for } k = 0, \\ 2 & \text{for } k > 0, \end{cases} \tag{10}$$

$$\tag{11}$$

and $\omega_0 = \frac{2\pi}{T_0}$ where T_0 is the period of the current output sequence. Using the vector notation

$$\vec{Y}_{ik} = \begin{bmatrix} Y^c_{ik} \\ Y^s_{ik} \end{bmatrix}, \tag{12}$$

the cell equations can be transformed into the Fourier domain, resulting in:

$$\vec{U}_{ik} = \vec{X}_{ik} + \sum_j w_{ij} \begin{bmatrix} \cos(k\omega_0 T_{ij}) & -\sin(k\omega_0 T_{ij}) \\ \sin(k\omega_0 T_{ij}) & \cos(k\omega_0 T_{ij}) \end{bmatrix} \cdot \vec{Y}_{jk}, \tag{13}$$

$$\vec{V}_{ik} = \frac{1}{1 + (k\omega_0\tau_v)^2} \begin{bmatrix} 1 & -k\omega_0\tau_v \\ k\omega_0\tau_v & 1 \end{bmatrix} \cdot \vec{U}_{ik}, \tag{14}$$

$$\vec{Y}_{ik} = \frac{\beta_k}{T_0}\int_0^{T_0} y_i(t) \begin{bmatrix} \cos(k\omega_0 t) \\ \sin(k\omega_0 t) \end{bmatrix} dt. \tag{15}$$

For linear cells with

$$y_i(t) = m_i v_i(t) \tag{16}$$

where m_i is a gain constant, the spectral cell output is simply

$$\vec{Y}_{ik} = m_i \vec{V}_{ik}. \tag{17}$$

Note that the transformation into the Fourier domain causes the time delays T_{ij} to become a simple quadrature phase matrix. Similarly, the cell time constant τ_v becomes an amplitude scaling factor that depends on the spectral frequency component $k\omega_0$.

The spectral error criterion on which the training is based is obtained by comparing the actual output sequence to the desired output sequence. In the time domain, the error as a function of time is

$$e_i(t) = y_i^{(d)}(t) - y_i(t) \tag{18}$$

where i refers only to the network output cells here. The total error for the current output sequence over all output cells (N_o) is given by

$$E = \frac{1}{T_0} \int_0^{T_0} \sum_{i=1}^{N_o} \frac{1}{2} e_i^2(t)\, dt. \tag{19}$$

In the same way, the total error over all output sequences is simply the sum of E for each sequence. Since this is a linear operation, the derivation below will be done as if there only was one desired output sequence. The results are then summed over all output sequences to obtain the complete error criterion.

Using Parseval's theorem, the error in Equation (19) can be approximated by

$$E \approx \sum_{i=1}^{N_o} \left(\frac{1}{2} \sum_{k=0}^{K} \left[E_{ik}^c\right]^2 + \left[E_{ik}^s\right]^2 \right) \tag{20}$$

where E_{ik}^c and E_{ik}^s are the Fourier series coefficients of the temporal error sequence defined in Equation (18). As in the conventional back-propagation algorithm, the weights and time delays are adapted according to the gradient descent driving term,

$$\tau_a^{(z)} \frac{dz_{ij}}{dt} = -\eta \frac{\partial E}{\partial z_{ij}} \equiv \Delta z_{ij} \tag{21}$$

where z_{ij} is either w_{ij} or T_{ij} and $\tau_a^{(z)}$ is the adaptation time constant.

The spectral cell errors can be defined as

$$\vec{\delta}_{ik} = - \begin{bmatrix} \dfrac{\partial E}{\partial V_{ik}^c} \\ \\ \dfrac{\partial E}{\partial V_{ik}^s} \end{bmatrix}, \tag{22}$$

allowing the driving term to be written as

$$\Delta z_{ij} = \eta \sum_{k=0}^{K} \left[\delta_{ik}^c \frac{\partial V_{ik}^c}{\partial z_{ij}} + \delta_{ik}^s \frac{\partial V_{ik}^s}{\partial z_{ij}} \right] \tag{23}$$

using the chain rule. Note that the spectral cell errors $\vec{\delta}_{ik}$ are independent of z_{ij}, the weight or time delay being adapted.

The term that is dependent upon z_{ij}, $\partial \vec{V}_{ik} / \partial z_{ij}$, can be derived from the spectral cell equations in (13) and (14). When z_{ij} is the weight w_{ij}, $\partial \vec{V}_{ik} / \partial w_{ij}$ is given by

$$\frac{\partial \vec{V}_{ik}}{\partial w_{ij}} = \begin{bmatrix} A_{1k}(T_{ij}) & -A_{2k}(T_{ij}) \\ \\ A_{2k}(T_{ij}) & A_{1k}(T_{ij}) \end{bmatrix} \begin{bmatrix} Y_{jk}^c \\ \\ Y_{jk}^s \end{bmatrix} \tag{24}$$

where

$$A_{1k}(T_{ij}) = \frac{1}{1+(k\omega_0\tau_v)^2}\left[\cos(k\omega_0 T_{ij}) - k\omega_0\tau_v \sin(k\omega_0 T_{ij})\right] \tag{25}$$

and

$$A_{2k}(T_{ij}) = \frac{1}{1+(k\omega_0\tau_v)^2}\left[\sin(k\omega_0 T_{ij}) + k\omega_0\tau_v \cos(k\omega_0 T_{ij})\right]. \tag{26}$$

These coefficients incorporate the effects of both the cell filter and the interconnect time delays. When z_{ij} is the time delay T_{ij}, $\partial\vec{V}_{ik}/\partial T_{ij}$ is given by

$$\frac{\partial \vec{V}_{ik}}{\partial T_{ij}} = -w_{ij}\begin{bmatrix} A_{1k}(T_{ij}) & -A_{2k}(T_{ij}) \\ A_{2k}(T_{ij}) & A_{1k}(T_{ij}) \end{bmatrix}\begin{bmatrix} dY^c_{jk} \\ dY^s_{jk} \end{bmatrix} \tag{27}$$

where

$$\begin{bmatrix} dY^c_{jk} \\ dY^s_{jk} \end{bmatrix} = d\vec{Y}_{jk} = \frac{\beta_k}{T_0}\int_0^{T_0} \frac{dy_j(t)}{dt}\begin{bmatrix} \cos(k\omega_0 t) \\ \sin(k\omega_0 t) \end{bmatrix} dt. \tag{28}$$

Given the expressions for $\partial\vec{V}_{ik}/\partial z_{ij}$, only the spectral cell errors need to be determined in order to compute the adaptation driving term Δz_{ij}. For an output cell,

$$\vec{\delta}_{ik} = \sum_{k_2=0}^{K} \vec{E}_{ik_2}\cdot\nabla_{\vec{V}_{ik}}\vec{Y}_{ik_2} \tag{29}$$

in general. If the output cell is linear so $y_i(t) = m_i v_i(t)$, the spectral cell error simplifies to

$$\vec{\delta}_{ik} = m_i\vec{E}_{ik} \tag{30}$$

where $\vec{E}_{ik}$ is the set of Fourier series components of the error sequence defined in Equation (18). However, if the output cell has a nonlinear output function S_i, the spectral components of $v_i(t)$ are spread across the frequency spectrum. This effect is captured by the term,

$$\nabla_{\vec{V}_{ik}}\vec{Y}_{ik_2} = \begin{bmatrix} \dfrac{\partial Y^c_{ik_2}}{\partial V^c_{ik}} & \dfrac{\partial Y^c_{ik_2}}{\partial V^s_{ik}} \\[2ex] \dfrac{\partial Y^s_{ik_2}}{\partial V^c_{ik}} & \dfrac{\partial Y^s_{ik_2}}{\partial V^s_{ik}} \end{bmatrix}. \tag{31}$$

The components of this matrix can be calculated using:

$$\frac{\partial Y^c_{ik_2}}{\partial V^c_{ik}} = \frac{\beta_{k_2}}{2}\left[\frac{Y'^c_{i|k-k_2|}}{\beta_{|k-k_2|}} + \frac{Y'^c_{i(k+k_2)}}{\beta_{(k+k_2)}}\right], \tag{32}$$

$$\frac{\partial Y^s_{ik_2}}{\partial V^s_{ik}} = \frac{\beta_{k_2}}{2}\left[\frac{Y'^c_{i|k-k_2|}}{\beta_{|k-k_2|}} - \frac{Y'^c_{i(k+k_2)}}{\beta_{(k+k_2)}}\right], \tag{33}$$

$$\frac{\partial Y^c_{ik_2}}{\partial V^s_{ik}} = \begin{cases} \dfrac{\beta_{k_2}}{2}\left[\dfrac{Y'^s_{i(k+k_2)}}{\beta_{(k+k_2)}} + \dfrac{Y'^s_{i(k-k_2)}}{\beta_{(k-k_2)}}\right] & \text{for } k \geq k_2, \\ \dfrac{\beta_{k_2}}{2}\left[\dfrac{Y'^s_{i(k+k_2)}}{\beta_{(k+k_2)}} - \dfrac{Y'^s_{i(k_2-k)}}{\beta_{(k_2-k)}}\right] & \text{for } k < k_2, \end{cases} \tag{34}$$

$$\frac{\partial Y^s_{ik_2}}{\partial V^c_{ik}} = \begin{cases} \dfrac{\beta_{k_2}}{2}\left[\dfrac{Y'^s_{i(k+k_2)}}{\beta_{(k+k_2)}} - \dfrac{Y'^s_{i(k-k_2)}}{\beta_{(k-k_2)}}\right] & \text{for } k \geq k_2, \\ \dfrac{\beta_{k_2}}{2}\left[\dfrac{Y'^s_{i(k+k_2)}}{\beta_{(k+k_2)}} + \dfrac{Y'^s_{i(k_2-k)}}{\beta_{(k_2-k)}}\right] & \text{for } k < k_2, \end{cases} \tag{35}$$

(36)

where

$$\vec{Y}'_{ik} = \frac{\beta_k}{T_0}\int_0^{T_0} S'_i\big(v_i(t)\big)\begin{bmatrix}\cos(k\omega_0 t)\\ \\ \sin(k\omega_0 t)\end{bmatrix} dt \tag{37}$$

and $S'_i(v) = dS_i(v)/dv$. Clearly, the nonlinear case involves much more computation than when the cells are linear.

For a hidden cell (i.e., one whose output is not an output of the network), the spectral cell errors can be computed using the recursive back-propagation relationship,

$$\begin{bmatrix}\delta^c_{jk}\\ \\ \delta^s_{jk}\end{bmatrix} = \sum_i w_{ij}\sum_{k_2=0}^{K}\begin{bmatrix}\dfrac{\partial Y^c_{jk_2}}{\partial V^c_{jk}} & \dfrac{\partial Y^c_{jk_2}}{\partial V^s_{jk}}\\ \\ \dfrac{\partial Y^s_{jk_2}}{\partial V^c_{jk}} & \dfrac{\partial Y^s_{jk_2}}{\partial V^s_{jk}}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}A_{1k_2}(T_{ij}) & A_{2k_2}(T_{ij})\\ \\ -A_{2k_2}(T_{ij}) & A_{1k_2}(T_{ij})\end{bmatrix}\begin{bmatrix}\delta^c_{ik_2}\\ \\ \delta^s_{ik_2}\end{bmatrix}. \tag{38}$$

When the hidden cell is linear, the back-propagation expression can be simplified to

$$\begin{bmatrix}\delta^c_{jk}\\ \\ \delta^s_{jk}\end{bmatrix} = m_j\sum_i w_{ij}\begin{bmatrix}A_{1k}(T_{ij}) & A_{2k}(T_{ij})\\ \\ -A_{2k}(T_{ij}) & A_{1k}(T_{ij})\end{bmatrix}\begin{bmatrix}\delta^c_{ik}\\ \\ \delta^s_{ik}\end{bmatrix}. \tag{39}$$

Using these expressions, the adaptation driving term Δz_{ij} can be computed for each adaptable interconnect.

3.3 The Training Process

A training epoch consists of cycling through all sets of input-output training sequences. This process is illustrated in Figure 6 for two training sequences. For each training set,

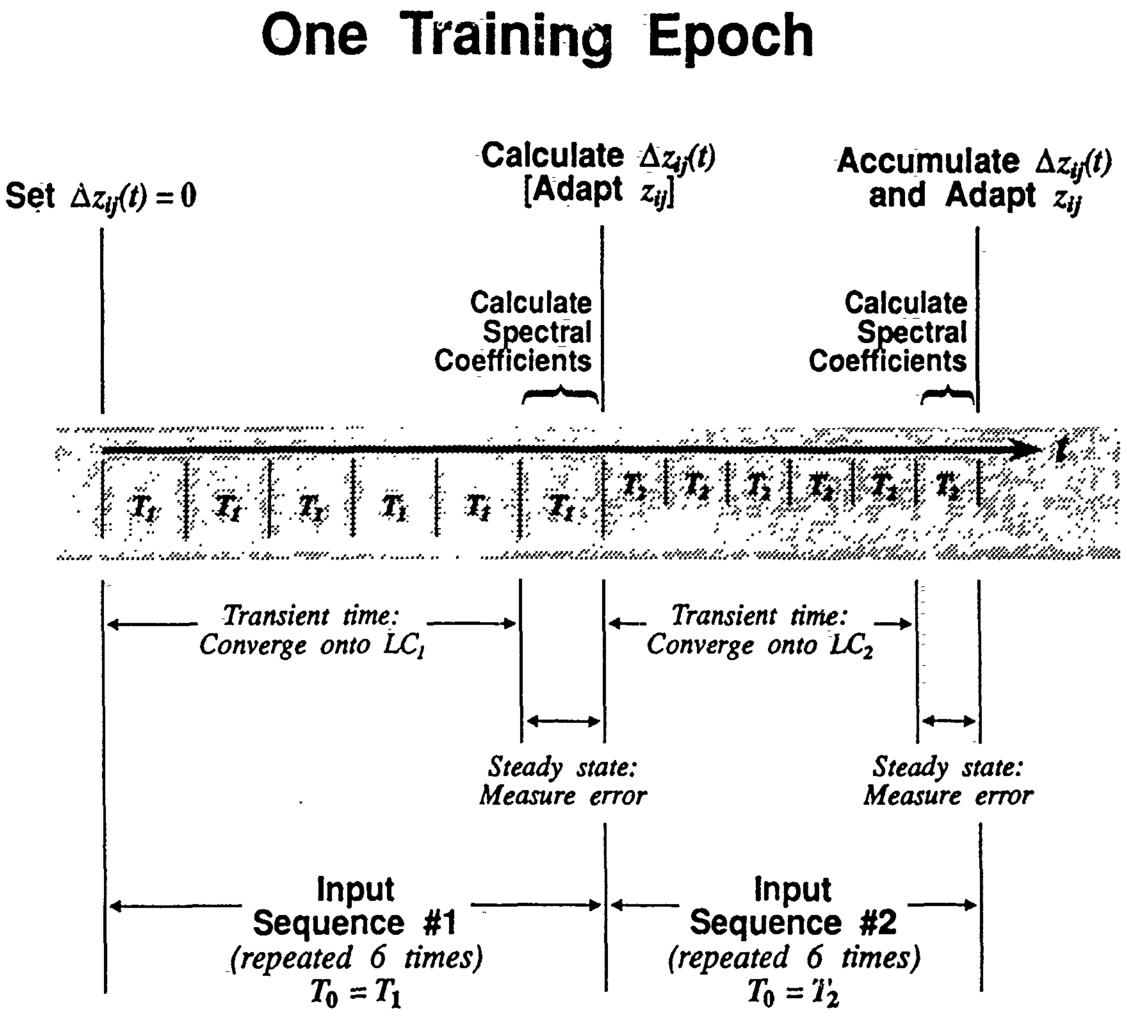


Figure 6: A training epoch.

the period T_0 is set to be the period of the current input and output sequence. Then, the input sequence is presented to the network n_e times in succession. The first $(n_e - 1)$ cycles provide the transient time during which all transients should decay away. During the $n_e{}^{\text{th}}$ cycle, the various Fourier series components are calculated and the gradient information in Δz_{ij} is accumulated. This process is repeated for each training set, at the end of which time, the weights and/or time delays are updated according to Equation (21).

If n_e is not large enough for transients to decay away, the spectral information computed during the last cycle will not be valid. The resulting adaptation probably will not converge onto a correct or even usable solution.

3.4 Performance Considerations

Several enhancements can be made to the SBP algorithm to decrease its average convergence time. First, the weights and time delays can be updated after each training set instead of after all training sets have been processed. Second, a momentum factor can be introduced into the adaptation driving term, producing a new driving term $\Delta z'_{ij}[t]$ that is given by

$$\Delta z'_{ij}[t] = \alpha_{\text{mom}} \Delta z'_{ij}[t-1] + (1 - \alpha_{\text{mom}}) \Delta z_{ij}[t] \tag{40}$$

where $\Delta z_{ij}[t]$ is Δz_{ij} at time step t and α_{mom} is the momentum factor. Typically, α_{mom} is small (0.2–0.3). The weights and time delays are then updated using

$$z_{ij}[t] = z_{ij}[t-1] + (1 - \alpha_a^{(z)}) \Delta z'_{ij}[t] \tag{41}$$

where $\alpha_a^{(z)} = e^{-1/\tau_a}$.

Another enhancement is the inclusion of variable adaptation gains, so η becomes $\eta_{ij}^{(z)}[t]$. That is, each weight and time delay has its own adaptive gain associated with it. We used the SuperSAB (Super Self-Adapting Back-propagation) method for adjusting these gains [2].

3.5 Discrete-Time Considerations

The discrete-time form for the cell equations is given in Equations (1)–(3). However, other approximations are necessary to simulate the SBP algorithm. First, the Fourier series components are computed using the approximation,

$$Y_{ik}^{c} = \frac{1}{T_0} \sum_{t=0}^{T_0-1} y[t] \cos(k\omega_0 t)\, dt, \tag{42}$$

$$Y_{ik}^{s} = \frac{1}{T_0} \sum_{t=0}^{T_0-1} y[t] \sin(k\omega_0 t)\, dt. \tag{43}$$

The time derivative dy/dt is approximated by the backward difference formula,

$$\frac{dy}{dt} \approx \Delta y[t] = y[t] - y[t-1]. \tag{44}$$

Finally, continuous time delays can be approximated by a linear interpolator between integral points. For example, the value of $y_j[t]$ at 5.3 iterations ago is approximated by

$$y_j[t-5.3] \approx (0.7)\, y_j[t-5] + (0.3)\, y_j[t-6]. \tag{45}$$

In general, T_{ij} can be decomposed into a integral part and a fractional part such that

$$T_{ij} = \lfloor T_{ij} \rfloor + \delta T_{ij} \tag{46}$$

where $\lfloor T_{ij} \rfloor$ is the largest integer less than or equal to T_{ij} (i.e., the truncation function) and δT_{ij} is the fractional offset ($0 \leq \delta T_{ij} < 1$). The general form for the linear interpolator is

$$y_j[t-T_{ij}] \approx \left\{ \left(1 - \delta T_{ij}\right) y_j\left[t - \lfloor T_{ij} \rfloor\right] \right\} + \left\{ \left(\delta T_{ij}\right) y_j\left[t - \lfloor T_{ij} \rfloor + 1\right] \right\}. \tag{47}$$

3.6 Choosing Parameters

The main parameters to select are n_e, the number of cycles per epoch for each training set, and K, the largest spectral component to compute. Several factors must be considered when choosing both parameters.

In order for the Fourier spectral analysis to be valid, the network must be in a dynamic steady state. Therefore, n_e should be large enough so all transients can decay away. However, the larger n_e is, the longer the runtime because each epoch takes longer to compute. Thus, the selection of n_e can be an iterative process. If the network is linear, an eigenvalue analysis can be performed for a reasonable set of weights and time delays to determine the longest time constant. Then, n_e initially can be set to, say, 4 times this duration. In general, though, a nonlinear network can be simulated for an initial set of weights and time delays and the response time can be measured. Then, n_e can be set to some multiple of this time constant. During the training process, the transient time can be measured to see if n_e can be increased or decreased. Ideally, an adaptive algorithm could be employed to adjust n_e automatically, although we did not experiment with any algorithms during the program.

By comparison, choosing K is much simpler, but the computation-speed tradeoff still exists. Ideally, K is as large as possible to represent the spectral information in the sequences. However, as K increases, the computation burder also increases, especially for nonlinear networks. A working guideline is that in order avoid aliasing when calculating the Fourier coefficients, K should be less than $T_0/10$ where T_0 is the period of the shortest training sequence. This limit also places an effective bandwidth limitation on the sequence to make it "smooth" and "slowly varying" in order to avoid aliasing with the spectral measurements, even when enough points are included in the calculation.

3.7 Time Delay Wrap-Around

When all training sequences have the same period T_0, the time delays T_{ij} can become "negative" by allowing them to wrap around 0 to T_0. Thus, if after an adaptation pass $T_{ij} < 0$, the actual time delay can be set to $T_0 + T_{ij}$. This wrap-around may lengthen the transient response so n_e must be large enough to accommodate this effect.

3.8 Sample Simulation Results

The simple recurrent network shown in Figure 7 was used to test the SBP algorithm on an IIR network. Both the weights w_{12} and w_{21} and the time delays T_{12} and T_{21} were allowed to be adaptable. The initial weights were set to 0.5 and the time delays to 0. The input interconnect was fixed with $w_{10} = 1$ and $T_{10} = 0$. The input sequence was

$$x_1[t] = \sin\left(\frac{2\pi t}{200}\right) \qquad (48)$$

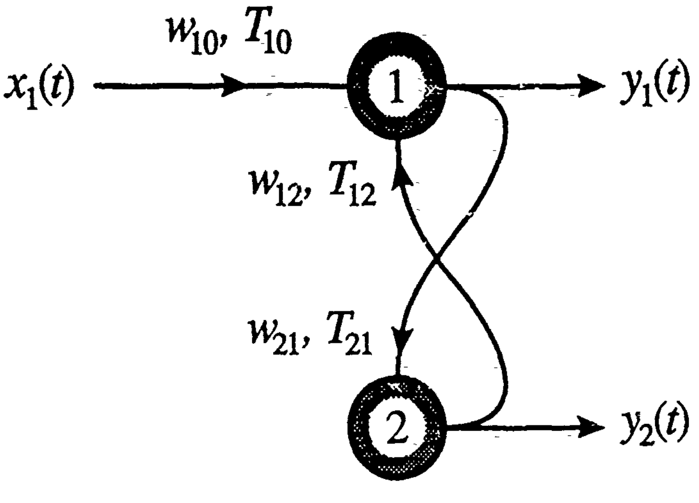


Figure 7: A simple linear filter/oscillator with an infinite impulse response (IIR).

so $T_0 = 200$ iterations. The desired output sequence was generated by the network by setting:

$$\begin{aligned} \tau_v &= 49.498, & \Rightarrow \alpha_v &= 0.98, \\ w_{10} &= 1, & T_{10} &= 0, \\ w_{21} &= 1, & T_{21} &= 30, \\ w_{12} &= -1, & T_{12} &= 30, \\ m_i &= 1. \end{aligned}$$

A total of five spectral components were computed, so $K = 4$. The number of cycles per epoch was set to $n_e = 2$, allowing for 1 transient cycle. The evolution of the training error E during the resulting adaptation is shown in Figure 8. The evolutions of the weights and the time delays are shown in Figure 9. The initial and final limit cycles are illustrated in Figure 10.

These plots show the successful adaptation performed by the SBP algorithm. The oscillations apparent in the evolution of the training error and the weights are caused by the gradient resets made by the SuperSAB adaptive gain algorithm. Since only one training sequence was used here, the time delays were allowed to wrap around 0 to T_0. The delay T_{12} takes advantage of this ability as illustrated in Figure 9(b). If the wrap-around is disabled, the adaptation settles into an unsatisfactory local minimum, thus preventing the training from completing successfully.

Other simulations showed that various combinations of the weights and time delays can be adapted. However, if the input interconnect $[w_{10}, T_{10}]$ is allowed to vary, the adaptation path is such that the network develops an eigenvalue that is very close to 1 in magnitude. The resulting long time constant prevents the network from reaching a steady state within the allocated time. Coincidentally, the adaptation converges to a solution but this solution is not correct because it cancels out the long transient response.

The SBP algorithm also has been demonstrated on the FIR network shown in Figure 11. The algorithm has trained successfully (1) the tap weights along the tapped delay line and (2) the tap time delays with the tap weights set to 1. These cases corresponding to training amplitude-only and phase-only FIR responses, respectively.

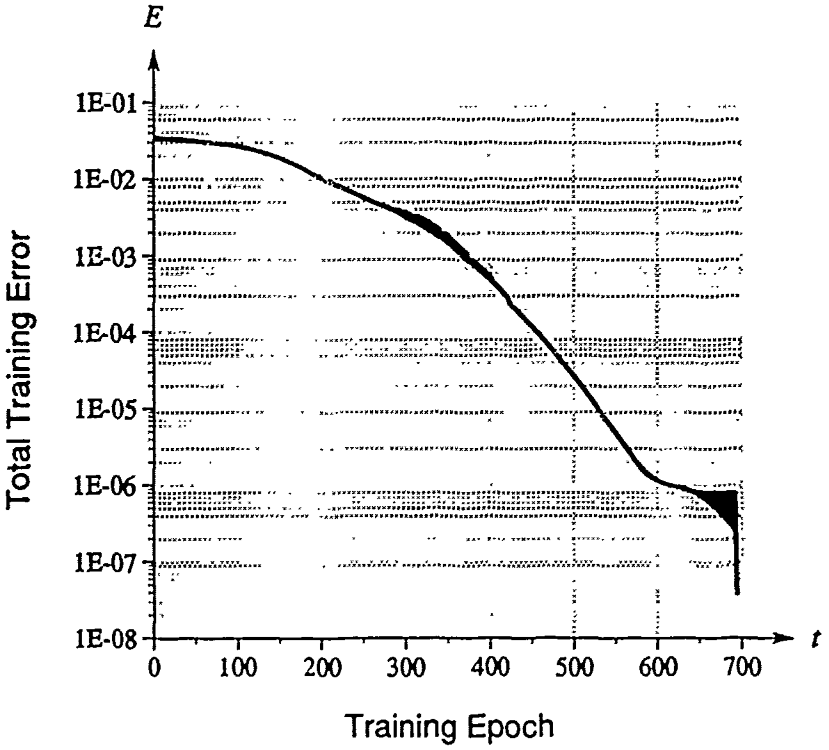


Figure 8: The evolution of the total training error versus the number of training epochs.

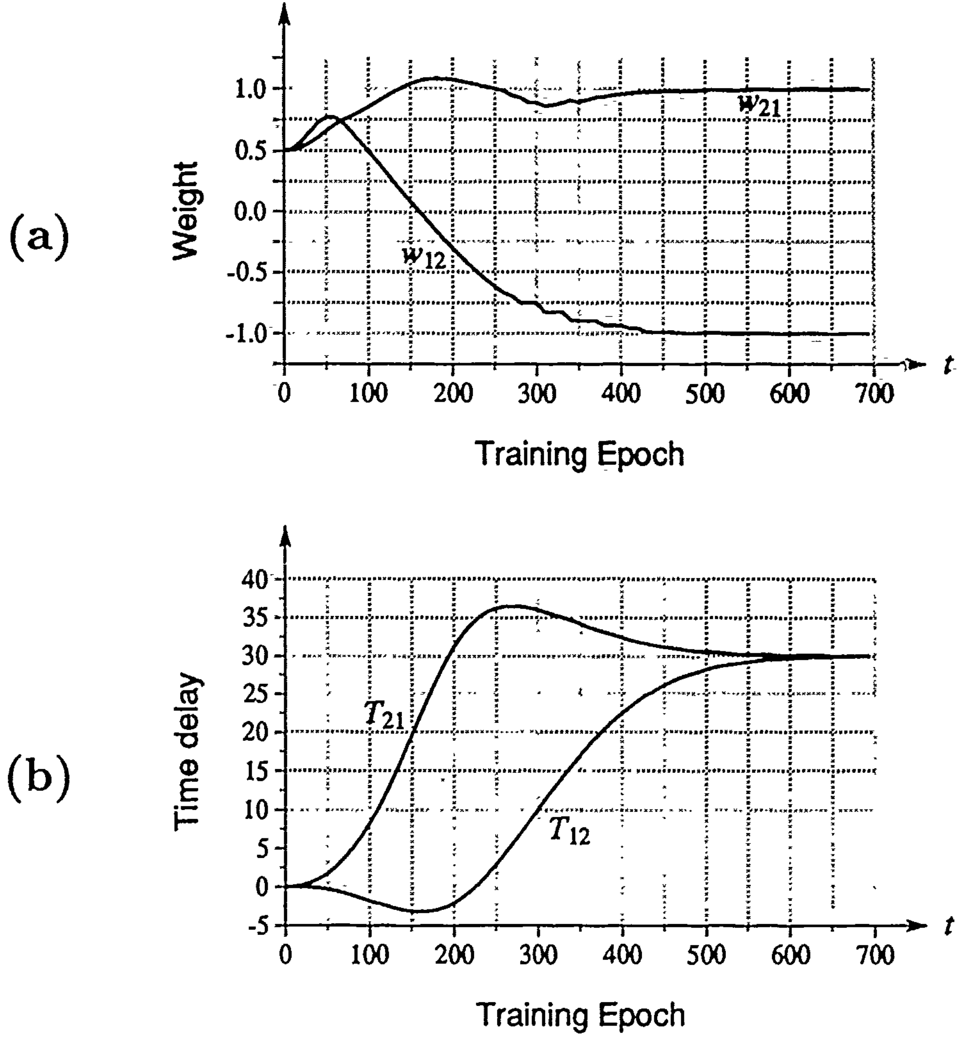


Figure 9: The evolution of (a) the weights and (b) the time delays. The negative delays indicate wrap-around has occurred. The actual delay is given by $T_{12} + T_0$ when $T_{12} < 0$.

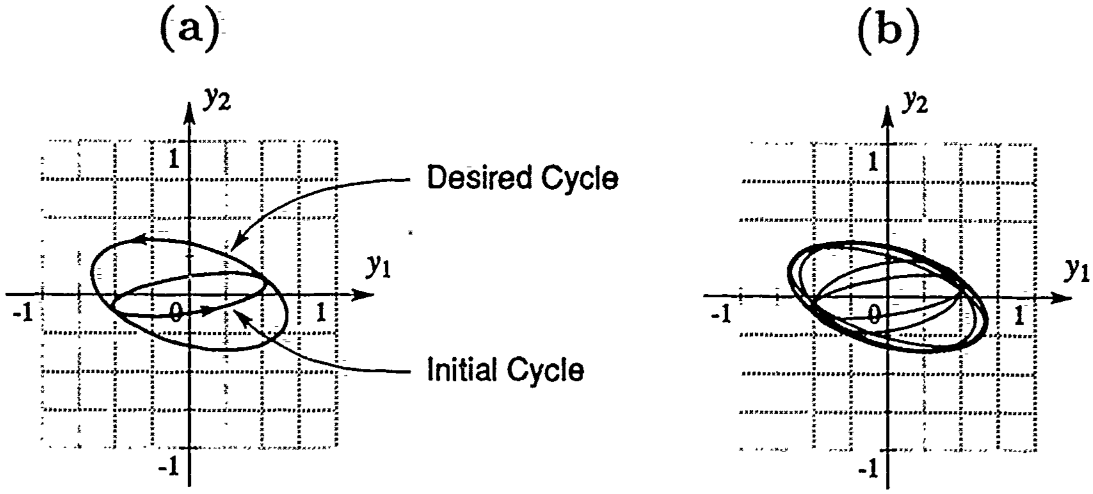


Figure 10: (a) The initial (before training) and desired (final) output limit cycles. (b) Snapshots of 8 output cycles taken during the training process with 100 epochs between cycles. This plot shows the output cycle evolution as w_{12}, w_{21}, T_{12}, and T_{21} are adapted.

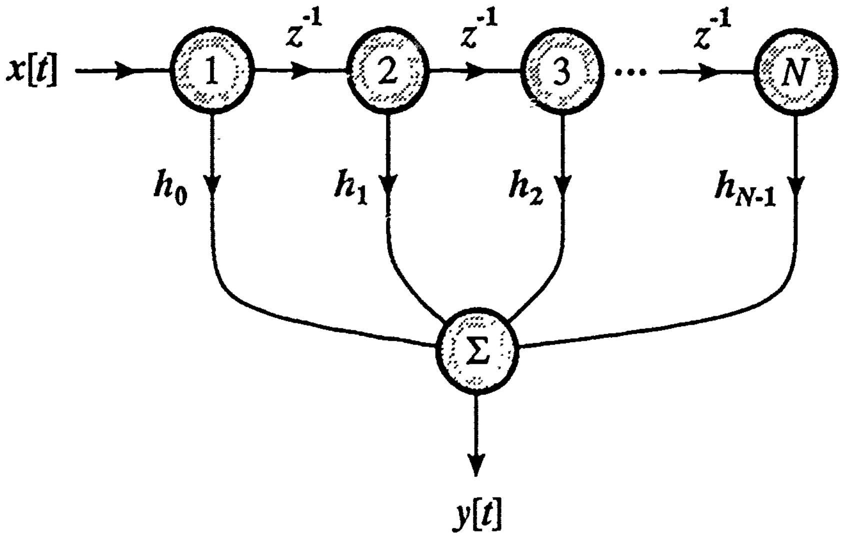


Figure 11: Basic structure of a finite impulse response (FIR) network of length N.

The amplitude-only case provides an indication of the strengths and weaknesses of the SBP algorithm. With a tapped delay line with $N = 25$ cells, the impulse response $h[t]$ shown in Figure 12(a) was used as the desired output sequence and the input sequence was a single unit impulse function. The total length of $h[t]$ was set to $T_0 = 100$ points to allow a large range of K values to be tested. The corresponding spectral coefficients for $h[t]$ are shown in Figures 12(b) and (c).

With $K = 6$, the SBP algorithm can train the FIR network so that only the first 7 (include $k = 0$) spectral components are matched. The resulting impulse response

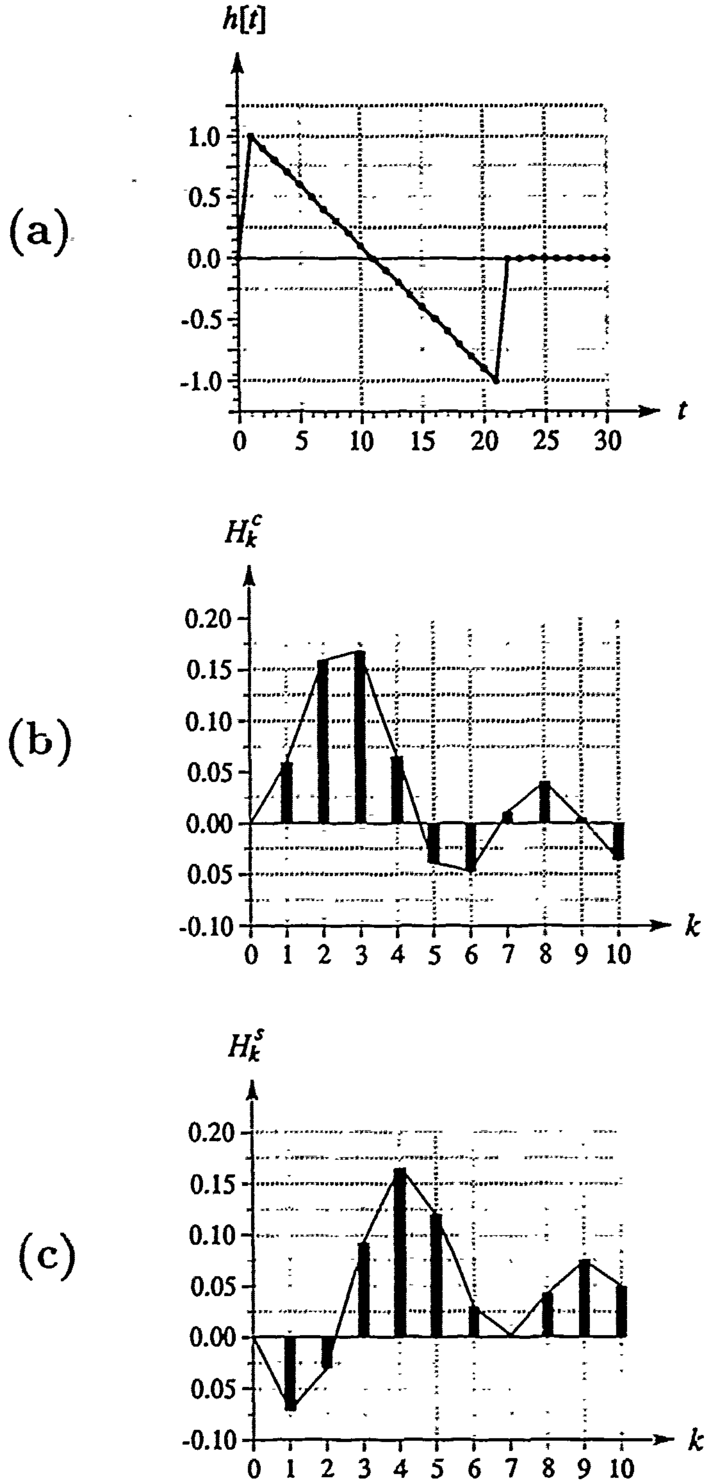


Figure 12: The desired impulse response for an amplitude-only FIR network and its first 10 Fourier series coefficients. (a) The first 30 points in $h[t]$. The remaining points (70) are zero. (b) The cosine spectral coefficients. (c) The sine spectral coefficients.

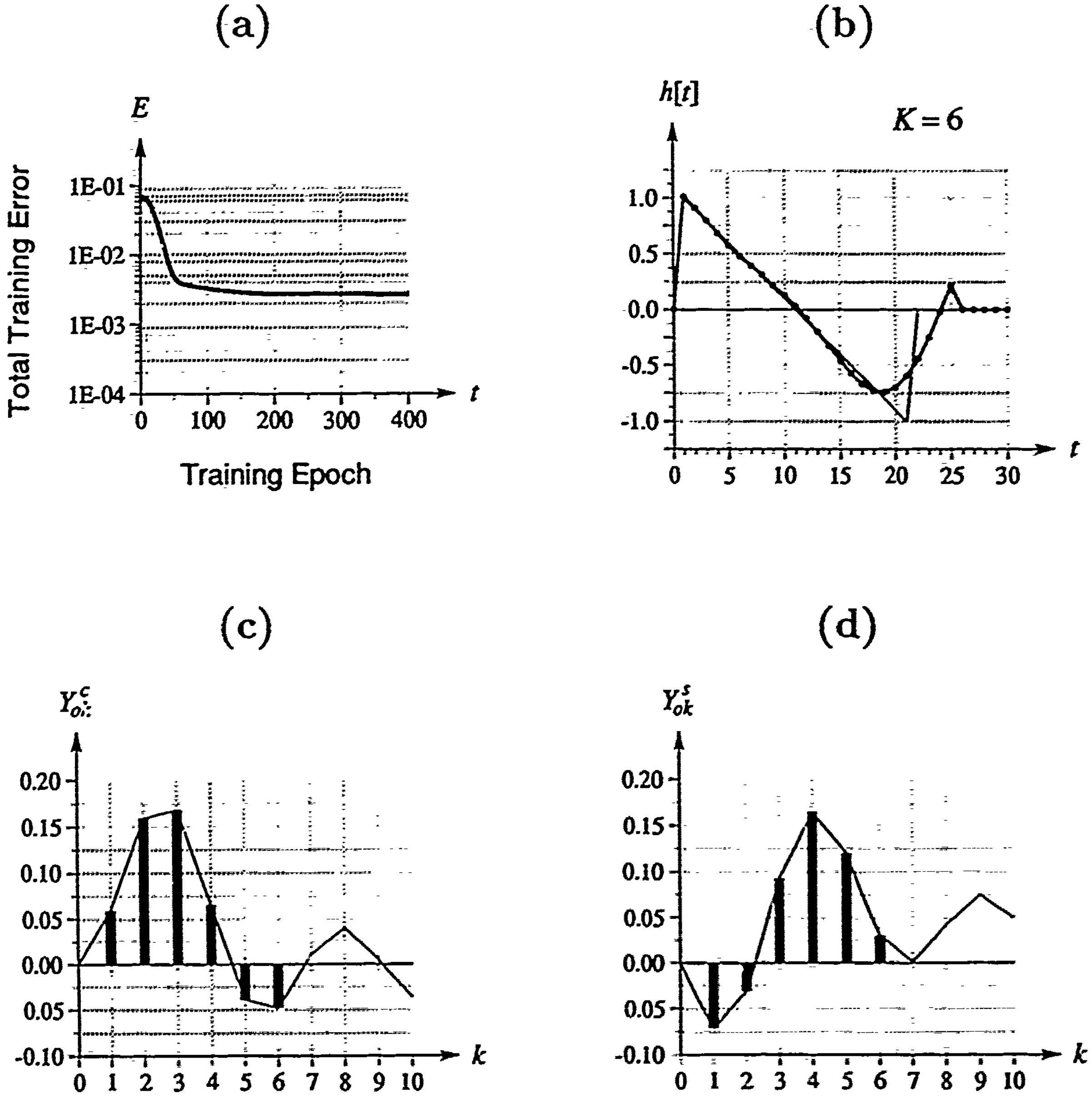


Figure 13: The learned impulse response for the FIR network when $K = 6$. (a) The evolution of the total training error. (b) The resulting impulse response as compared to the desired $h[t]$. (c) The cosine spectral coefficients of the actual $h[t]$. (d) The corresponding sine spectral coefficients.

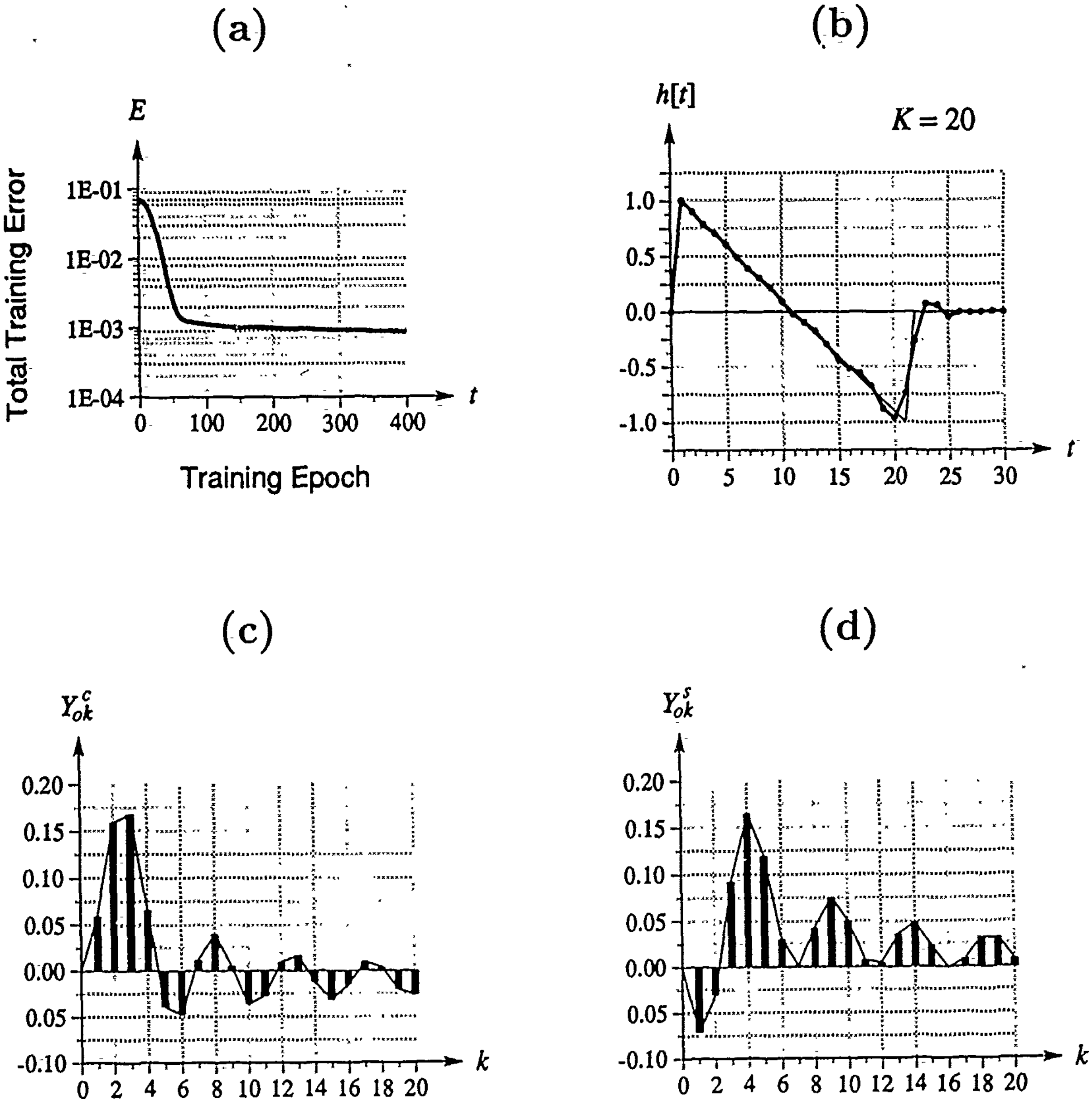


Figure 14: The learned impulse response for the FIR network when $K = 20$. (a) The evolution of the total training error. (b) The resulting impulse response as compared to the desired $h[t]$. (c) The cosine spectral coefficients of the actual $h[t]$. (d) The corresponding sine spectral coefficients.

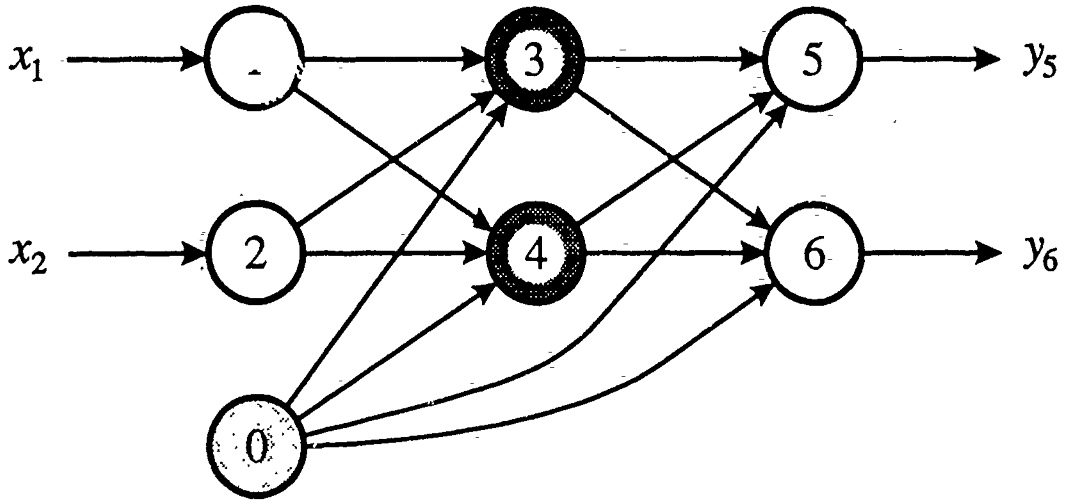


Figure 15: Dual-output XOR network. Cells 1, 2, 5, and 6 are linear whereas cells 3 and 4 have nonlinear output functions. Cell 0 has a constant output. All time delays are zero.

is close to the desired shape but the trailing edge is not reproduced faithfully. When K is increased to 20, the impulse response is much closer to the desired $h[t]$, including the trailing edge. However, when K is set to 24, there are only $T_0/K = 100/24 \approx 4$ points per period of the sampling cosine and sine functions for computing the spectral coefficients. This sampling is too infrequent and results in an aliasing condition. The adaptation with $K = 24$ fails to converge and instead causes the tap weights to grow without bound.

In a phase-only FIR network, the SBP algorithm can train either the tap time delays or those in the delay line. However, simulations showed that the trained FIR networks did not have the minimum time delays necessary to implement the filter. Thus, some post-processing of the resulting time delays would be necessary to realize a minimum-phase phase-only FIR network.

The SBP algorithm also was successful in training a network to learn static patterns. As an example, the network in Figure 15 was trained to learn the dual-output exclusive-OR (XOR) function described by the mapping:

Input Pattern	→	Output Pattern
x_1 x_2		y_5 y_6
0 0		0 1
0 1		1 0
1 0		1 0
1 1		0 1

The output y_5 is the XOR output and y_6 is its complement.

With a feedforward network of this type, the number of cycles per epoch (n_e) should be set to the number of layers including the input layer. Thus, n_e was set to 3 for this network. The input layer should be included in this count because the spectral coefficients for dy_i/dt need to be zero for static training patterns.

3.9 Disadvantages of the SBP Algorithm

The disadvantages of the SBP algorithm with respect to the conventional back-propagation algorithm are:

1. It is more demanding computationally. There is more overhead required to store and maintain the spectral information. Furthermore, the back-propagation process involves a vector quantity instead of a scalar quantity.

2. Convergence can be slower. Training information is not collected at every time step but only after the entire output sequence has been observed.

3. A steady state network solution must exist for the training to be successful. If the network output is still on its transient response or is chaotic, the spectral information will be invalid and will cause the adaptation to fail.

4. Arbitrarily shaped sequences cannot be trained because of aliasing concerns. The discrete-time Fourier series of the sequences must be computable without any aliasing problems.

4 Limit Cycle Finite State Machine

4.1 Overview

A block diagram of a limit cycle finite state machine (LC-FSM) is shown in Figure 16. It consists of a memory with many addressable limit cycles (e.g., a SONN) and a controller to govern the transitions from one cycle to another. The cycles produced by the memory correspond to the logical FSM states. The inputs and outputs to the LC-FSM are intended to be continuous-time limit cycles also, but fixed-point inputs and outputs are possible.

Like a traditional FSM, the operation of the LC-FSM is governed by a state transition diagram. Each transition is determined by the current-state cycle from the memory, the input cycle, *and* the relative phase between them. Thus, for the same state and input cycles, several different state transitions are possible by assigning each transition to a different relative phase. The total number of possible transitions is limited by the resolution of the transition detectors, the phase-measuring components of the cycle transition controller. These detectors will be trained using the SBP algorithm to recognize the desired transition conditions.

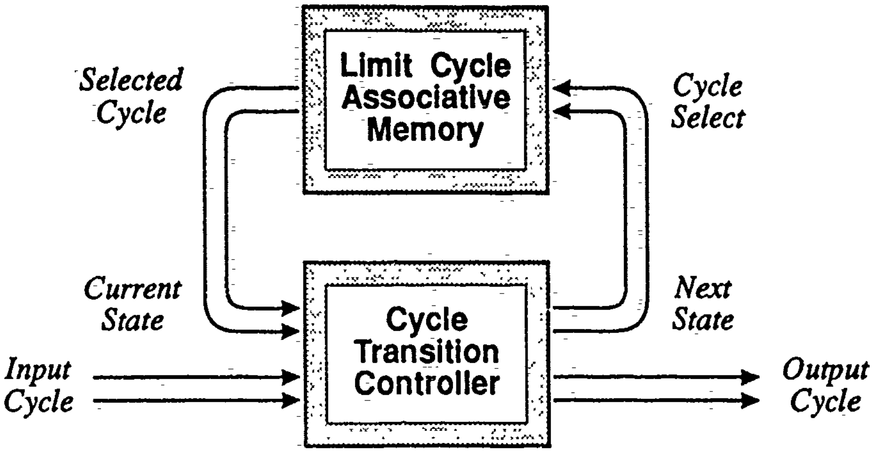


Figure 16: Block diagram of a finite state machine based on limit cycles (LC-FSM).

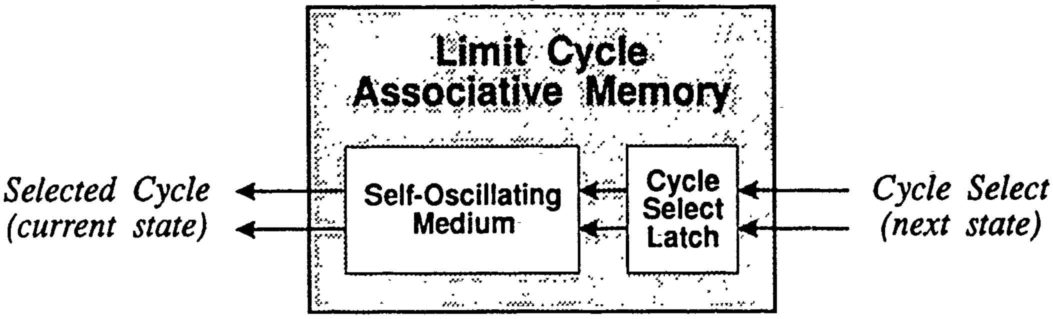

Limit Cycle Associative Memory
Selected Cycle (current state)
Self-Oscillating Medium
Cycle Select Latch
Cycle Select (next state)

Figure 17: Expanded block diagram of the LC-FSM memory.

4.2 Limit Cycle Associative Memory

The function of the memory in the LC-FSM is to store the current state of the machine. An expanded block diagram of this component is shown in Figure 17. It has two components, a latch and a self-oscillating medium. The function of the latch is to accept, recognize, and then store the current memory input if the recognition is successful. The latch output is a fixed point corresponding to the cycle to be selected. The self-oscillating medium produces a unique oscillation for a unique input. With the output of the latch driving it, the self-oscillating medium thus generates a unique limit cycle as the memory output for the most recently recognized memory input. Since the latch as a simple storage register, the main component of the limit cycle associative memory is the self-oscillating medium. The SONN model presented in Section 2 is useful in this role.

4.3 Cycle Transition Controller

A detailed block diagram of the cycle transition controller is shown in Figure 18. It performs two tasks: (1) to detect desired transitions and request the next state from the memory, and (2) to generate the desired output cycles and/or vectors.

The goal of the first task is to transform a multidimensional oscillatory signal (two limit cycles) and produce a single binary-like signal indicating the input has been "recognized." This signal then can be used to trigger a transition to a new state. The transformation should be sensitive to the amplitudes and phases of the two limit cycles (i.e., their shapes and relative phase). This task is the function of the transition detectors (TDs) and the in-phase recognizers (IPRs) in Figure 18.

A transition detector network is shown in Figure 19. The network accepts two cycles as its inputs, the input cycle to the LC-FSM and the current-state cycle from the memory SONN. Cells D1 and D2 are linear and respond immediately, so $\tau_v = 0$ and $y_i = v_i = u_i$ where i is either D1 or D2 here. Cell D0 is a constant-output cell with $y_{D0} = 1$ and provides a source of trainable biases for the other two cells. All the weights and time delays are trainable, except for the time delays associated with cell D0 which are fixed at 0.

Because a transition detector contains only a single layer of linear cells, the training process using the SBP algorithm is very quick, particularly when the SuperSAB adaptive gain algorithm is enabled. To learn a given set of input and current-state cycles, these cycles are presented to the network at the desired relative phase and the output is trained to be the function,

$$y_1[t] = y_2[t] = \frac{1}{2}\left[1 + \sin\left(\frac{2\pi t}{T_0}\right)\right], \tag{49}$$

where T_0 is the period of *both* cycles. This signal generates a linear limit cycle as shown in Figure 20(a) and indicates a recognized oscillatory state. In order for the linear network in Figure 19 to work, the input and current state cycles must have the same period, but this period can vary from one to another transition detector. Note, however, that either the input or the current-state (but not both) could be constant and not a cycle. Furthermorc, the phase of the sinusoid in Equation (49) can be set arbitrarily.

The second component in the recognition process is the in-phase recognizier, shown in Figure 21. This network consists of two heterodyne circuits in parallel. For both circuits, cells I1-I4, B1-B4, M1-M4, and O1 and O2 are linear ($y_i = v_i$) and respond instantaneously ($\tau_v = 0$). However, cells M1-M4 have a multiplicative input structure (as opposed to the conventional additive form) such that

$$u_i[t] = \prod_{j=1}^{N_i} w_{ij} y_j[t - T_{ij}] \tag{50}$$

where i is one of M1-M4 and j is one of either I1-I4 or B1-B4, whichever is appropriate. Only the interconnects between cells O1 and O2 and M1-M4 are trainable. Both the weights and the time delays are allowed to be trained. Cells O1 and O2 do <u>not</u> have trainable biases as this would destroy the recognition properties of the network. Essentially, the two heterodyne circuits create all possible cross products of the input with

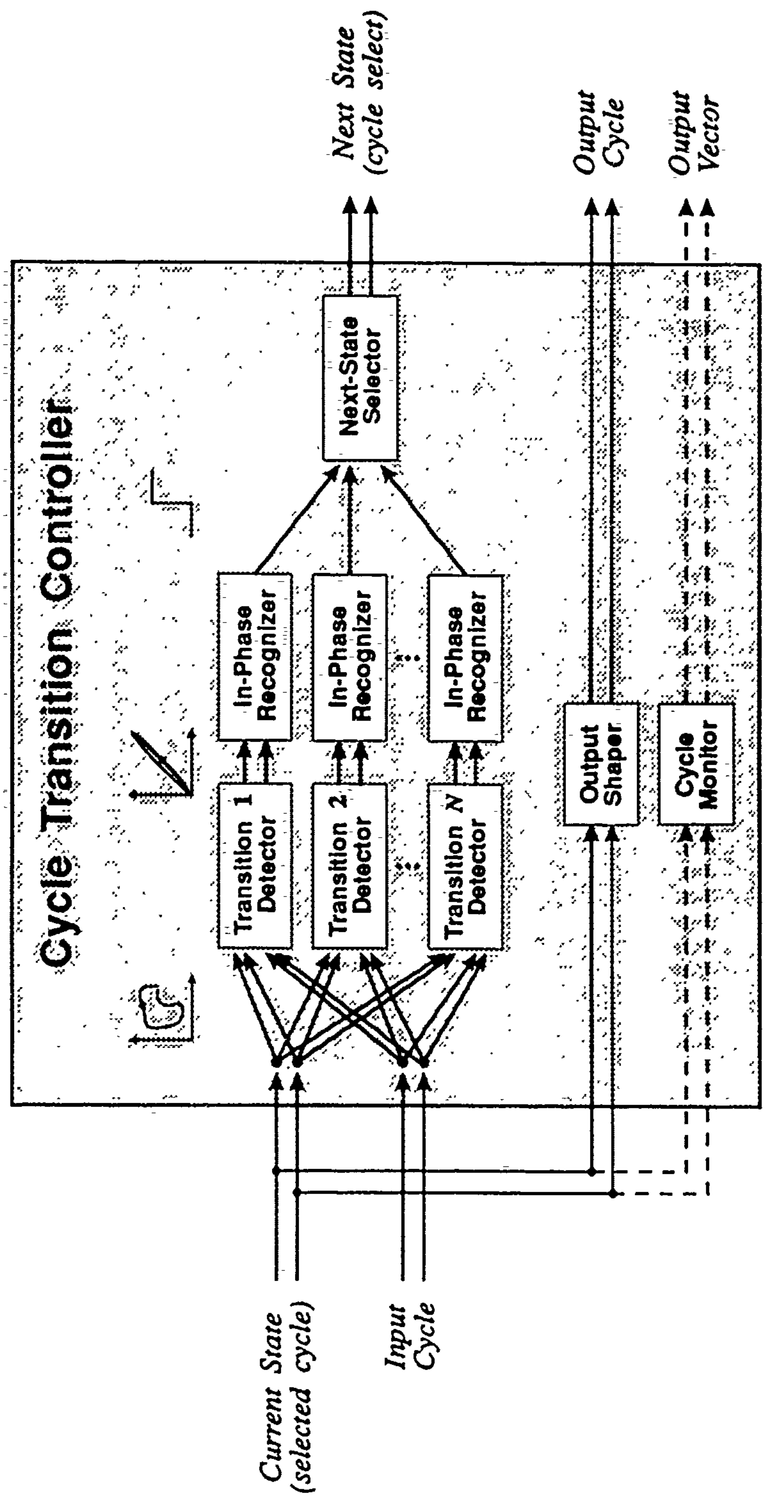


Figure 18: Expanded block diagram of the cycle transition controller in the LC-FSM. The cycle monitor is an optional component and thus has dashed input and output arrows.

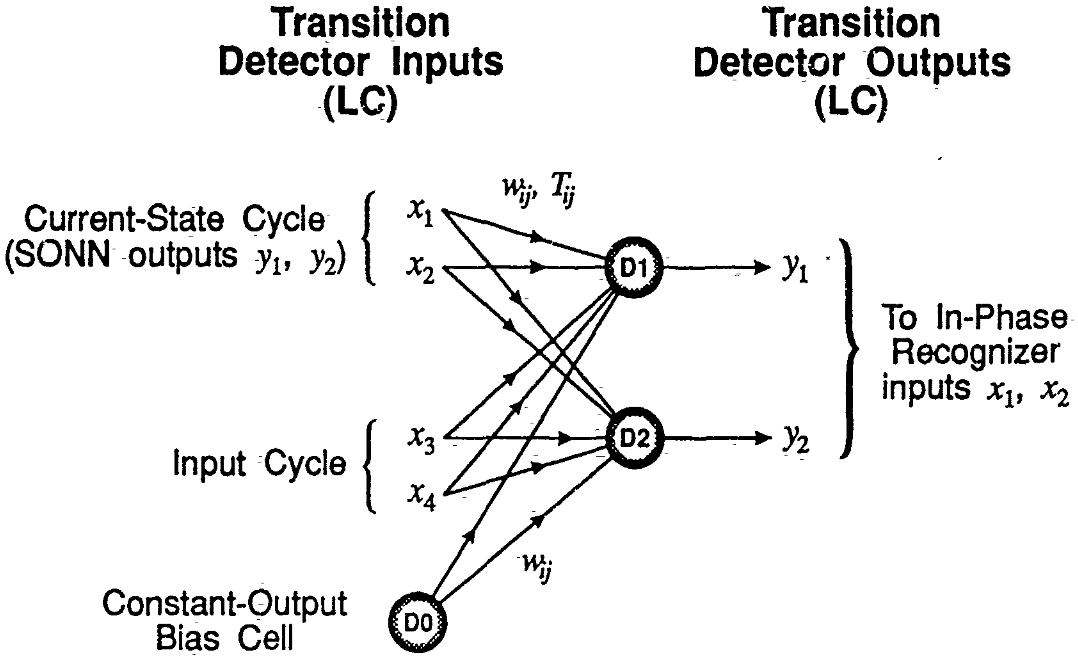

Transition Detector Inputs (LC)
Transition Detector Outputs (LC)
Current-State Cycle (SONN outputs y_1, y_2)
x_1
x_2
w_{ij}, T_{ij}
D1
y_1
To In-Phase Recognizer inputs x_1, x_2
Input Cycle
x_3
x_4
D2
y_2
Constant-Output Bias Cell
D0
w_{ij}

Figure 19: A transition detector network for the LC-FSM. All weights and time delays are trainable here using the SBP algorithm, except for the time delays associated with the constant-output cell. These delays are fixed at $T_{ij} = 0$.

the input shifted by −0.5. Using the SBP algorithm, the intermediate outputs y_1 and y_2 are trained to be 1 for all time when the input is the linear limit cycle described by Equation (49). The SBP algorithm effectively combines the cross-products to cancel out the oscillatory terms and leave only the DC value.

Cells G1, G2, and O3 are used to combine the outputs of the two heterodyne circuits. Cells G1 and G2 have Gaussian-shaped output functions in the form,

$$S_i(v_i) = e^{-m(v_i-1)^2}, \tag{51}$$

to select the region of the (y_1, y_2) space about the point $(1, 1)$. The gain was set to $m = 50$ and the cell filters were activated with $\tau_v = 20$ iterations. The finite bandwidth prevents trajectories passing over $(1, 1)$ from being falsely recognized as being the desired linear limit cycle.

The actual threshold point for a decision is made by cell O3 using the sum of the outputs of G1 and G2. Cell O3 is a thresholding cell with finite bandwidth ($\tau_v = 20$ iterations) to inhibit any output oscillations about the threshold point $v_0 = 0.8$.

The final output of the complete in-phase recognizer (y_3) is a binary signal that is 1 when the input is the linear limit cycle described by Equation (49). Because its function is common to all transition detectors, the in-phase recognizer needs to be trained only once and then replicated for each TD. When combined with a transition detector, the TD/IPR combination produces a 1 every time a transition condition (i.e., the right input and current-state cycles) is present.

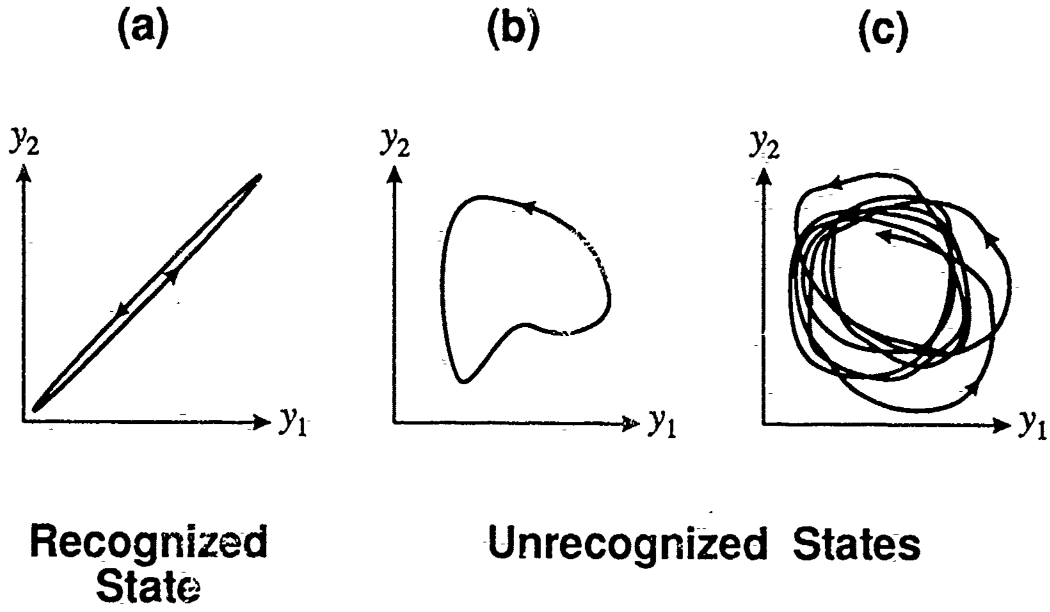


Figure 20: Possible outputs for a transition detector. Only the near-linear cycle shown in (a) is to be considered a recognized state. Then, the input cycle and the current-state cycle to t' is particular transition detector are amplitude- and phase-matched, signifying a condition for a state transition has been recognized. The spatially-distributed cycle in (b) and the chaotic trajectory in (c) indicate no state transition condition is present.

4.4 Next-State Controller

The outputs of all TD/IPRs go to the next-state controller as shown in Figure 18. This network is a simple vector-to-vector mapping network which forms the correspondence between the detected transition condition and the next state. Its function is to realize the state transition diagram of the LC-FSM.

4.5 Simulated LC-FSM Results

A LC-FSM based on the SONN presented in Section 2 was simulated to test the capabilities of the TD/IPR networks. Although the SONN contains many cycles, only the ones corresponding to the constant inputs $(x_1, x_2) = (0,1), (1,0)$, and $(1,1)$ are considered here. The cycles generated by these inputs will be referred to as **01**, **10**, and **11**, respectively. This SONN will be referred to as the memory SONN. For simplicity, the output of the LC-FSM is simply taken to be the current state of the memory SONN.

In addition to the memory SONN, another identical SONN was used as the input to the simulated LC-FSM. It is referred to as the input SONN. Before going into the LC-FSM, the output of the input SONN is sent through a variable-length time delay to allow the relative phase of the input SONN cycles to be adjusted manually with respect to the current-state cycle from the memory SONN.

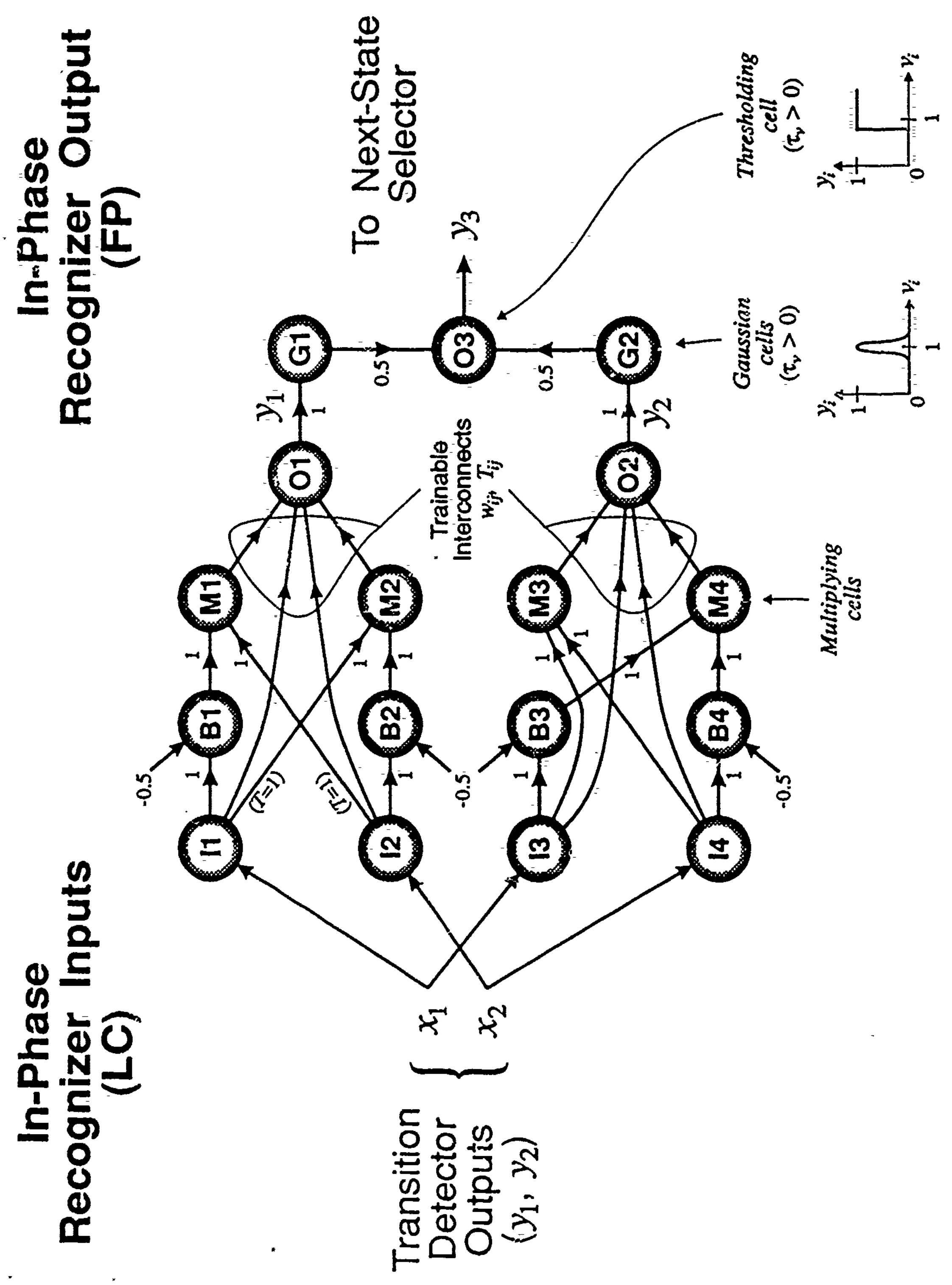

In-Phase Recognizer Inputs (LC)
In-Phase Recognizer Output (FP)
Transition Detector Outputs (y_1, y_2)
x_1
x_2
I1
I2
I3
I4
B1
B2
B3
B4
M1
M2
M3
M4
O1
O2
O3
G1
G2
-0.5
-0.5
-0.5
-0.5
1
(T=1)
(T=1)
Trainable Interconnects w_{ij}, T_{ij}
Multiplying cells
y_1
y_2
0.5
0.5
y_3
To Next-State Selector
Gaussian cells ($\tau_v > 0$)
Thresholding cell ($\tau_v > 0$)
y_i
v_i
0
1

Figure 21: An in-phase recognizer network. The "−0.5" inputs come from a constant-output bias cell such as D0 in Figure 19. All time delays are 0 except where labeled ($T = 1$). These delays are needed to compensate for the unit propagation delay through cells B1 and B2. Redundant cells B3 and B4 are shown for clarity to illustrate the two parallel heterodyne circuits.

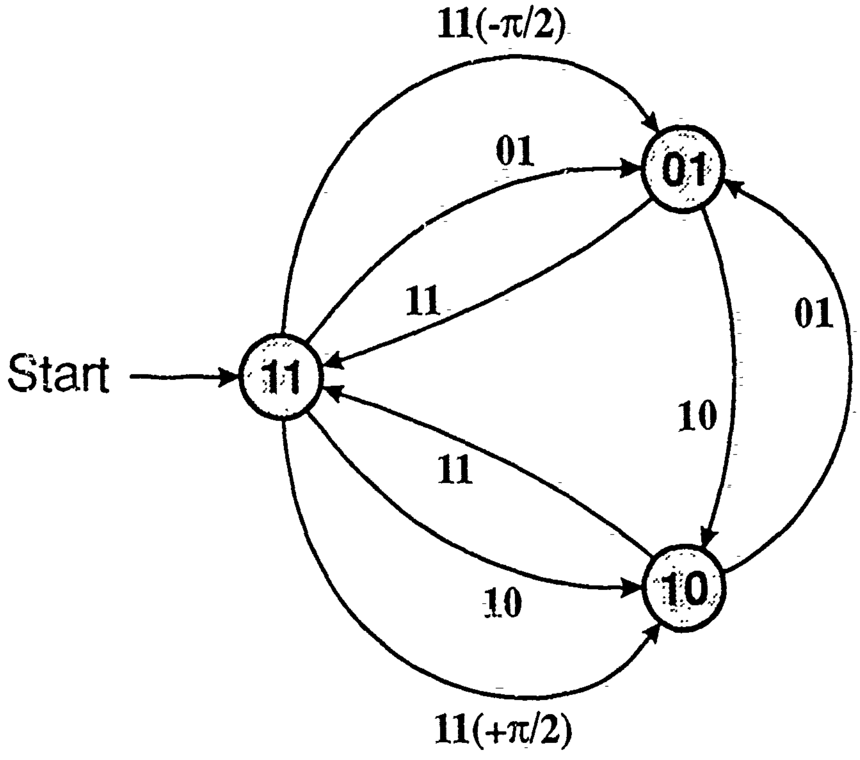


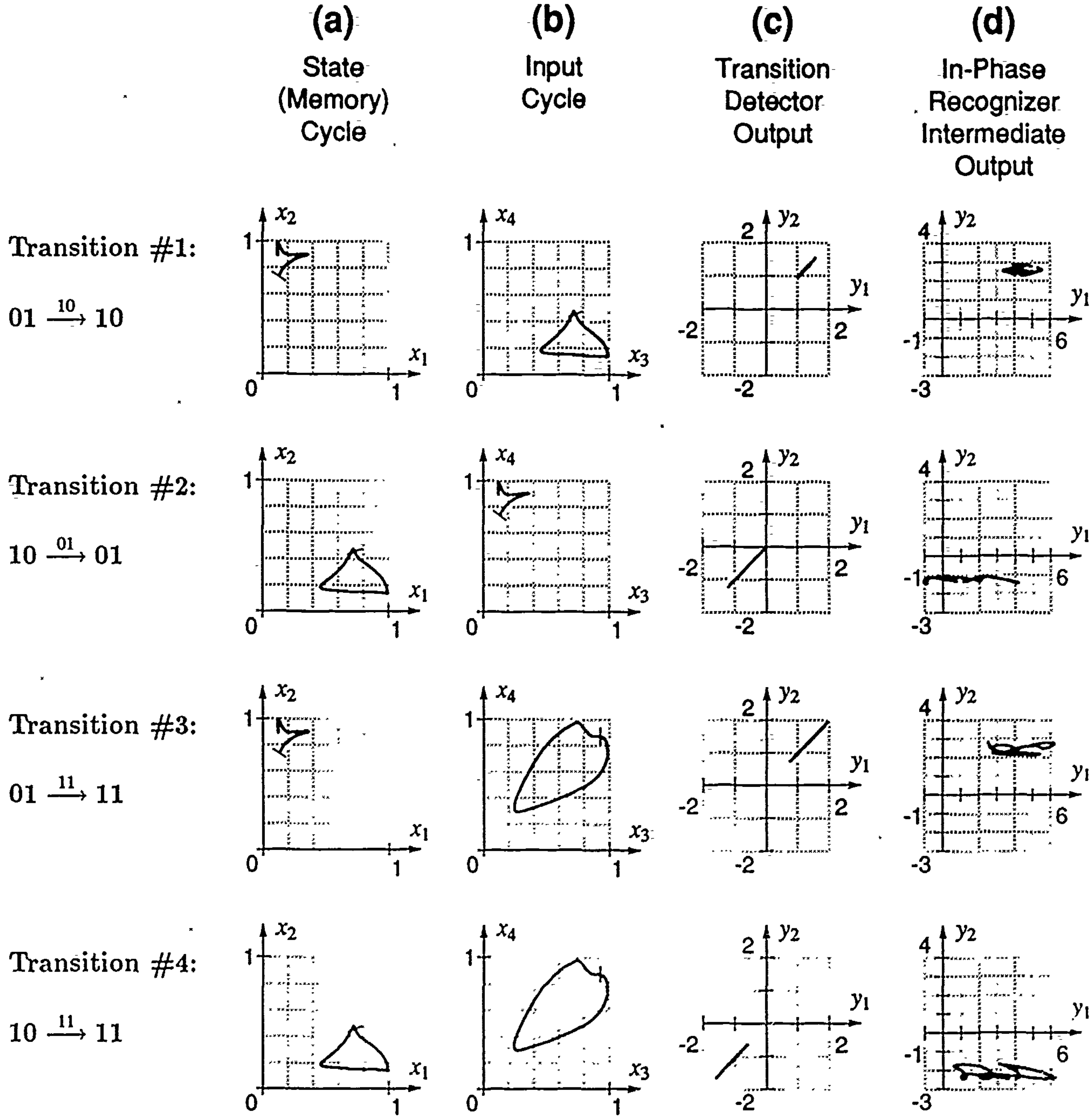
Figure 22: State transition diagram for the simulated LC-FSM. The ($\pm\pi/2$) notation denotes a relative phase shift with respect to the current-state cycle.

The simulated LC-FSM implements the state transition diagram shown in Figure 22. The LC-FSM states correspond to the memory SONN cycles and the transitions are labeled with the input SONN cycles. Thus, if the memory SONN is oscillating in the state 01 and the input SONN is oscillating in the state 10 *at the correct phase*, the LC-FSM should make a transition to the 10 state in the memory SONN. The actual SONN cycles used and their relative phases are shown in Figure 23(a)–(b).

To demonstrate the phase sensitivity of the transition detection, two transitions from the 11 state are induced by the 11 input cycle with two different relative phases. If the input cycle leads the current-state cycle by $+\pi/2$, the transition is to the 10 state. If, on the other hand, the input cycle lags the current-state cycle by $-\pi/2$, the state changes to 01. Since the period of the SONN oscillations is about 64 iterations, a phase lead/lag of $\pm\pi/2$ equals a relative advance/delay of 16 iterations for the input cycle with respect to the current-state cycle.

As an example of the TD/IPR networks in operation, consider the transition $11 \xrightarrow{01}$ 01. The response of the transition detector for this transition to all 8 transition conditions is shown in Figure 23(c). For the corresponding in-phase recognizer, the intermediate output at cells O1 and O2 is shown in Figure 23(d). A composite view of the IPR graphs is shown in Figure 24.

The discrimination capabilities of the combination of the TD/IPR networks now



continued

Figure 23: The response of the transition detector and in-phase recognizer for the transition 11 $\xrightarrow{01}$ 01 to all 8 transition conditions. The end-bar on each cycle indicates the starting point for the cycle. Transition #5 is the one by which this transition is recognized. (a) The LC-FSM current-state cycle. (b) The LC-FSM input cycle. (c) The output of the 11 $\xrightarrow{01}$ 01 transition detector. (d) The intermediate output (y_1, y_2) of the corresponding in-phase recognizer at cells O1 and O2. These plots are typical of the other transitions. Continued on the next page.

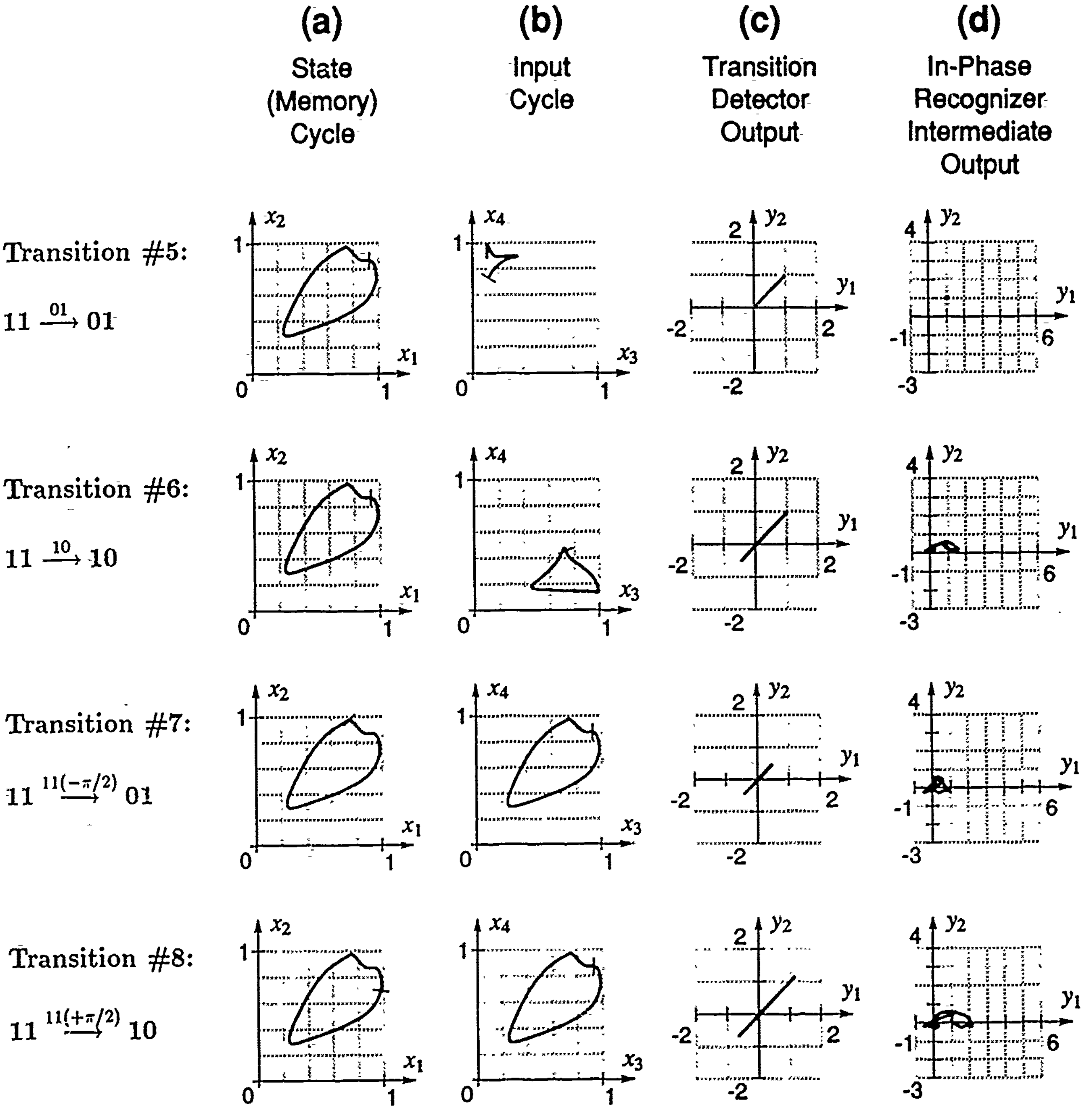


Figure 23 continued.

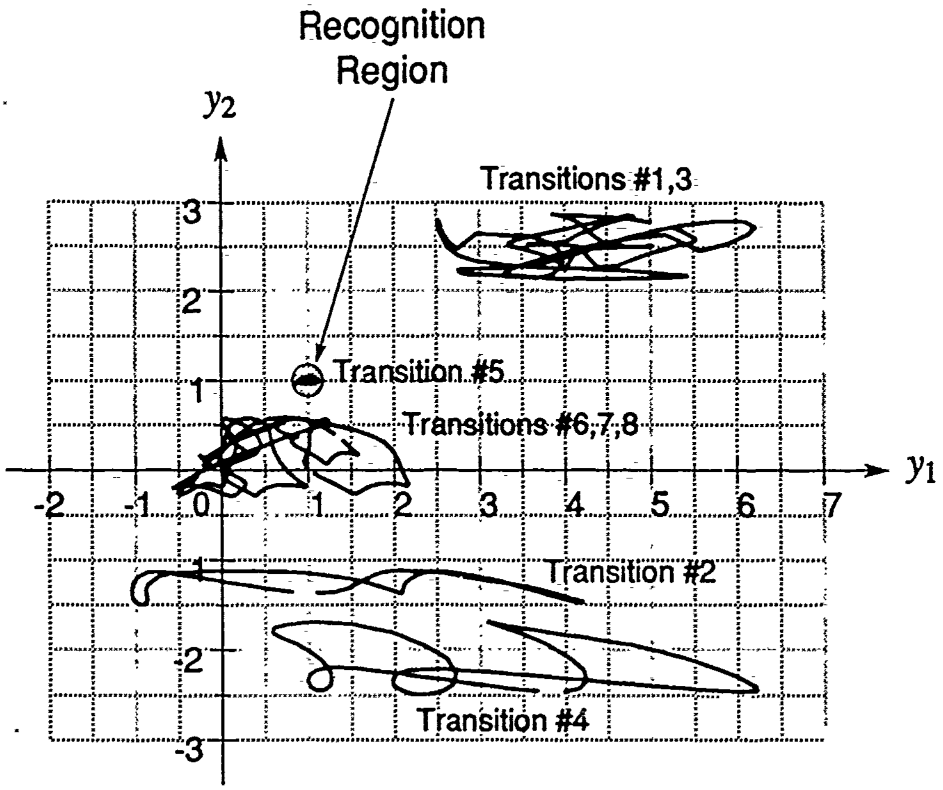


Figure 24: Composite view of the in-phase recognizer output for the transition $11 \xrightarrow{01} 01$ to all 8 transition conditions. This plot is combination of the graphs shown in Figure 23(d). The shaded circular region around (1,1) is the recognition region whereby a trajectory contained within indicates a transition condition is present and accepted.

become apparent. The outputs of the TD mostly are shifted and scaled versions of the desired linear limit cycle. However, the IPR separates these cycles into distinct parts of the output space defined by cells O1 and O2. It thus becomes an easy task to decide if the LC-FSM input cycle and current-state cycle correspond to a valid transition condition. The cycle must stay close to the point $(y_1, y_2) = (1, 1)$. With the parameters chosen for the TD/IPR networks, the cycle recognition is done in under 3 periods and has a relative phase sensitivity of about 7°.

This particular transition (#5, $11 \xrightarrow{01} 01$) was chosed because it was the worst case of the 8 possible transitions. In the other cases, the IPR outputs were more spread out, making the discrimination even easier.

Given the set of 8 binary output signals from the TD/IPR networks, the next-state selector network was designed to set and reset the latch for the memory SONN such that the state transition diagram shown in Figure 22 was realized. The resulting network is shown in Figure 25 and its function table is given in Table 1.

The latch network for the memory SONN is simply a pair of identical set-reset (SR) flip-flops and is shown in Figure 26. Each flip-flop has two inputs, one for setting the output to be 1 and the other for resetting the output to 0. The latching property

Table 1: Input-output relationship for the next-state selector network shown in Figure 25. The subscripts on the inputs correspond to the transition numbers given in Figure 23.

Inputs from In-Phase Recognizers (Transition Number)								Outputs to Latch			
								To Flip-Flop 1		To Flip-Flop 2	
								(Set)	(Reset)	(Set)	(Reset)
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y_1	y_2	y_3	y_4
1	0	0	0	0	0	0	0	1	−1	−1	1
0	1	0	0	0	0	0	0	−1	1	1	−1
0	0	1	0	0	0	0	0	1	−1	1	−1
0	0	0	1	0	0	0	0	1	−1	1	−1
0	0	0	0	1	0	0	0	−1	1	1	−1
0	0	0	0	0	1	0	0	1	−1	−1	1
0	0	0	0	0	0	1	0	−1	1	1	−1
0	0	0	0	0	0	0	1	1	−1	−1	1

Table 2: Function table for the latch in Figure 26.

Set	Reset	Output
x_1, x_3	x_2, x_4	y_1, y_2
0	0	hold
1	0	1
0	1	0
1	1	1 (unstable)

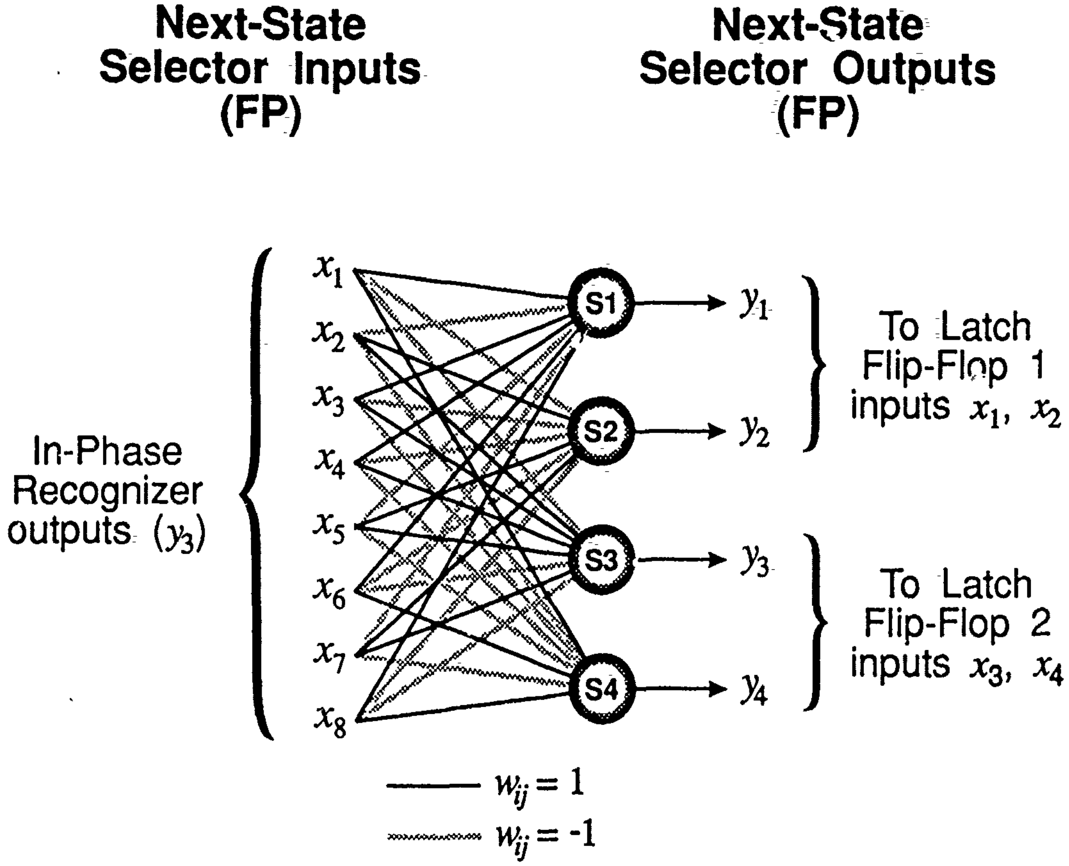

Next-State Selector Inputs (FP)
Next-State Selector Outputs (FP)
In-Phase Recognizer outputs (y_3)
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
S1
S2
S3
S4
y_1
y_2
y_3
y_4
To Latch Flip-Flop 1 inputs x_1, x_2
To Latch Flip-Flop 2 inputs x_3, x_4
$w_{ij} = 1$
$w_{ij} = -1$

Figure 25: Next-state selector network. The weights are given in Table 1. There are no time delays here.

comes from the self-feedback connection on each cell (with a weight of 1) and the lateral inhibitory connections. There are no time delays in any of the interconnects, nor is any training required. The cells respond instantaneously ($\tau_v = 0$) and have a binary (on/off) output function described by

$$S_i(v_i) = \begin{cases} 1 & \text{for } v_i >= v_0, \\ 0 & \text{for } v_i < v_0, \end{cases} \tag{52}$$

where the threshold $v_0 = 0.9$ and i is either L1 or L2 here. The resulting function table for each flip flop is given in Table 2.

A block diagram of the complete simulated LC-FSM is shown in Figure 27. It illustrates the component networks and their relative interconnections. The only addition is an adjustable time delay T_D so the relative phase of the input limit cycle with respect to the current-state (memory) cycle can be varied. This ability is needed to establish each of the 8 transition conditions because of the high phase sensitivity of the TD/IPR networks.

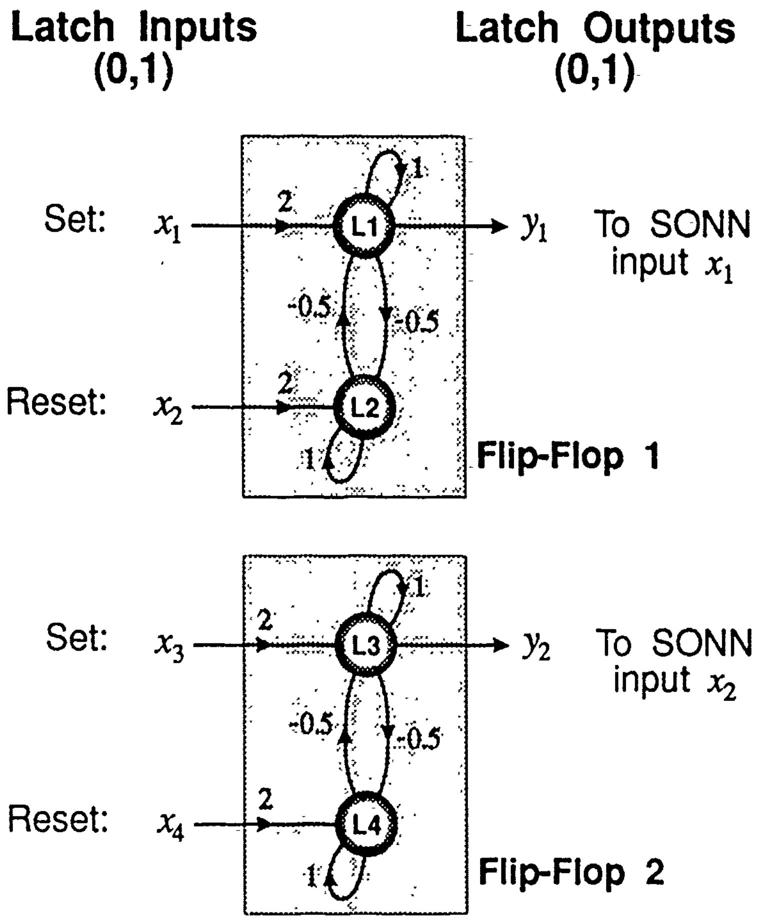


Figure 26: The latch network for the associative memory in the LC-FSM. It consists of two independent flip-flop networks, one for each input to the memory SONN.

Table 3: Schedule for changing the input cycle delay T_D and the resulting LC-FSM state changes. The transition reference numbers (e.g., #4) are taken from Figure 23. The times at which the transitions occurred are taken to be when the latch outputs changed.

At Time t	T_D Set To	Relative Input Phase	Transition Occurred At	Transition Simulated	New LC-FSM State
0	0	0°	—	—	**11**
400	40	225°	593	#8	**10**
800	0	0°	897	#4	**11**
1200	2	11°	1310	#7	**01**
1600	43	242°	1808	#3	**11**

Before the simulation run, the memory latch outputs were set to ($y_1 = 1, y_2 = 1$) and the memory SONN was run for 200 iterations to allow it to converge onto the **11** cycle. Then, the LC-FSM was simulated for 2400 iterations, during which time the input SONN was set to the **11** state. Only the phase of this cycle with respect to the memory SONN cycle was adjustable via T_D. The schedule on which T_D was changed is given in Table 3. The state evolution of the LC-FSM during this run is shown in Figure 28. For reference, the times at which T_D was changed and when the resulting state transitions occurred are shown with shaded vertical lines.

In this figure, the temporal trace and various state-space snapshots are shown for both the input cycle to the LC-FSM as delayed by T_D and the LC-FSM output as taken from the memory SONN. Each state-space plots consists of the 64 points before the reference arrow above the top (y_1) trace.

At the bottom of the figure, the corresponding evolutions of the TD/IPR networks along with the memory latch outputs are shown to illustrate the timing relationship between all the signals. The TD/IPR trace actually is a superposition of all 8 TD/IPR outputs. Similarly, the latch trace is a superposition of both y_1 and y_2 latch outputs.

Several observations can be made from Figure 28. First, the transitions occurred within 3 periods of the input limit cycle as predicted by the TD/IPR discussion. Next, the TD/IPR signals are usually very short in duration. Once the latch outputs have changed, the memory SONN changes its output cycle, thus destroying the previous transition condition. Finally, the first transition **11** $\xrightarrow{11(+\pi/2)}$ **10** and the third transition **11** $\xrightarrow{11(-\pi/2)}$ **01** show that the same input cycle can stimulate different transitions based on the relative phase with respect to the memory cycle.

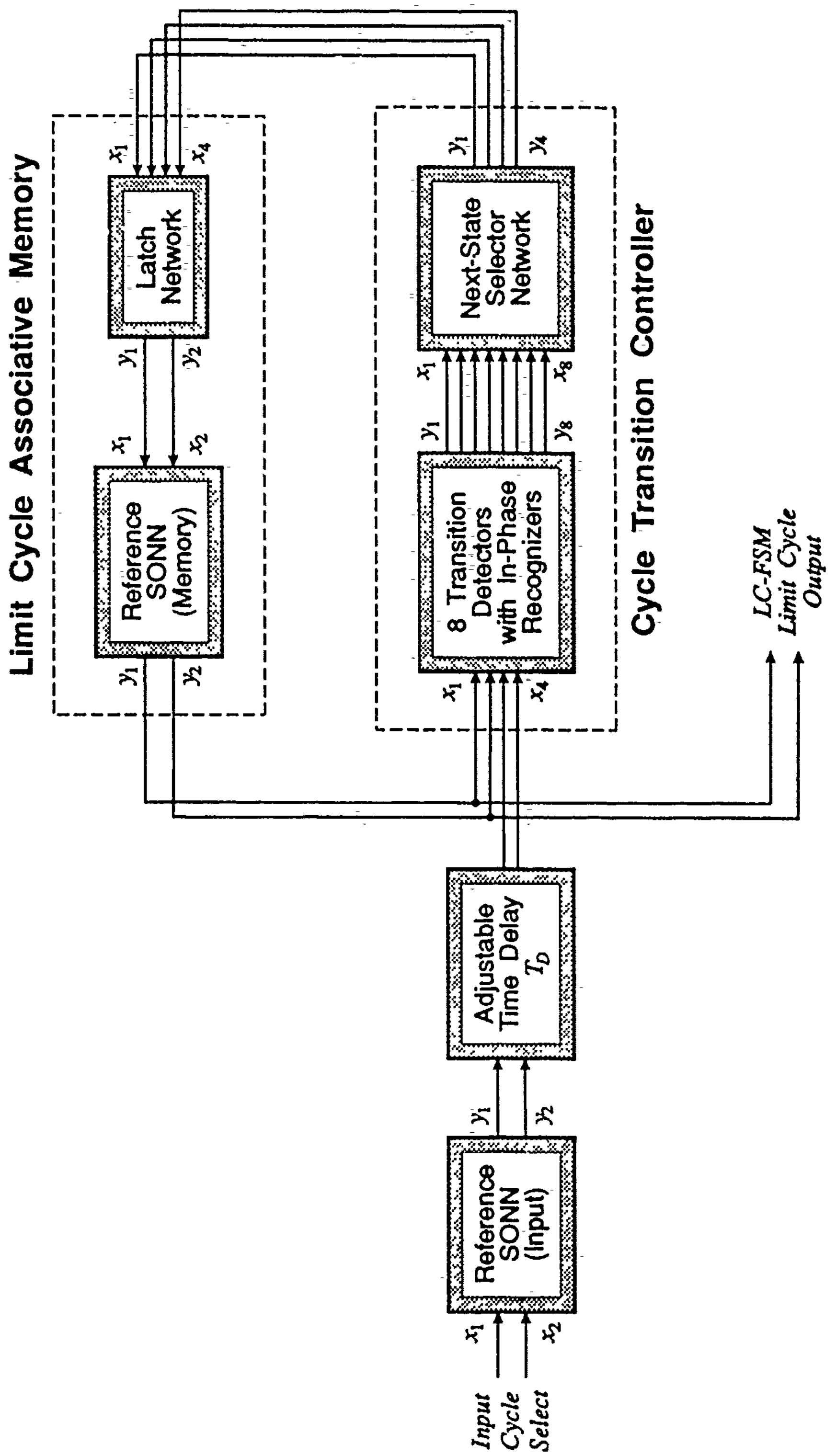

Limit Cycle Associative Memory
Latch Network
Reference SONN (Memory)
Cycle Transition Controller
Next-State Selector Network
8 Transition Detectors with In-Phase Recognizers
Adjustable Time Delay T_D
Reference SONN (Input)
Input Cycle Select
LC-FSM Limit Cycle Output
x_1
x_2
x_4
x_8
y_1
y_2
y_4
y_8

Figure 27: Block diagram illustrating the interconnection of the component networks in the simulated LC-FSM.

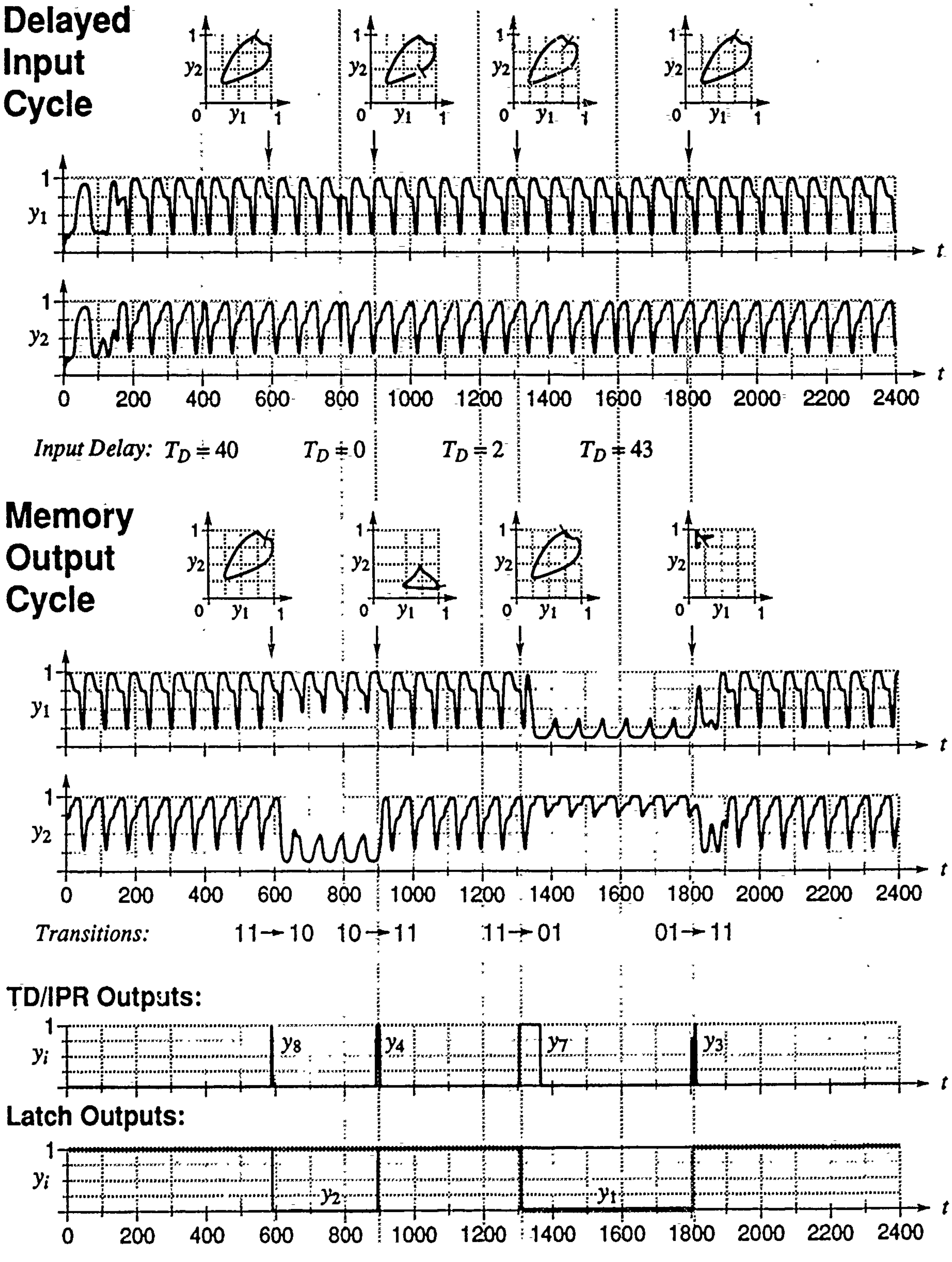


Figure 28: State progression of the simulated LC-FSM.

5 Optical Implementation Considerations

5.1 Optical SONN Architecture

The SONN model discussed in Section 2 and shown in Figure 1 on page 11 was designed with an optical implementation in mind. The SONN has the following desirable features:

1. All cells have the same nominal output characteristic. SLMs with sine-squared or parabolic output functions effectively match the nominal sigmoid curve shown in Figure 4 on page 13.

2. The off-center, on-surround canonical interconnect topology within the levels simplifies the interconnect scheme and provides a low diversity of interconnects to realize.

3. The model is very tolerant of static parameter variations (easily ±20%). These variations can arise in several ways. First, the crosstalk within the interconnects can appear as weight perturbations. A nonuniform readout beam effectively changes $y_{\max}$, the maximum cell output value, over many cells. Similarly, nonuniformities within the SLM can cause the same effect, along with variations in the gain m of the output function.

An optical architecture for the SONN depicted in Figure 1 is shown in Figure 29. It is based on a ring of optically-addressed SLMs (O-SLMs) connected via interconnection holograms (IHs). The O-SLMs implement the cell functions and are sequenced by a computer controller. In Figure 29, O-SLMn contains the cells for the n^{th} layer in each level. For example, the cell arrangement for O-SLM1 is shown in Figure 30. This device has the cells from the first layer in all three levels. Similarly, the second and third layers of each level are on O-SLM2 and O-SLM3, respectively.

The IHs are fixed, computer-generated holograms which realize both the interlevel and the intralevel interconnects. The interlevel connections are made by IH1. Its interconnect mapping between the cells on O-SLM1 and O-SLM3 is shown in Figure 31. The off-center, on-surround canonical interconnect topology in the levels is implemented by IH2 and IH3. This intralevel mapping is shown in Figure 32.

The input to the SONN is obtained from an electrically-addressed spatial light modulator (E-SLM) that is driven by the computer controller. Hologram IH1 performs the input mapping from the input cells to the first layer in the first level. This mapping is shown in Figure 33 for the case when the four input cells in the first level are independent (in contrast to Figure 1 where they are derived from two external input cells).

The SONN output is collected by the controller using a two-dimensional photodetection device such as a camera or a photodetector array. This output is derived from the last layer in the first level. Hologram IH4 forms the corresponding mapping from O-SLM3 to the camera and is illustrated in Figure 34.

This architecture implements the discrete-time approximation of the continuous-time network equations used in Section 3 with no time delays in the interconnects. In order to

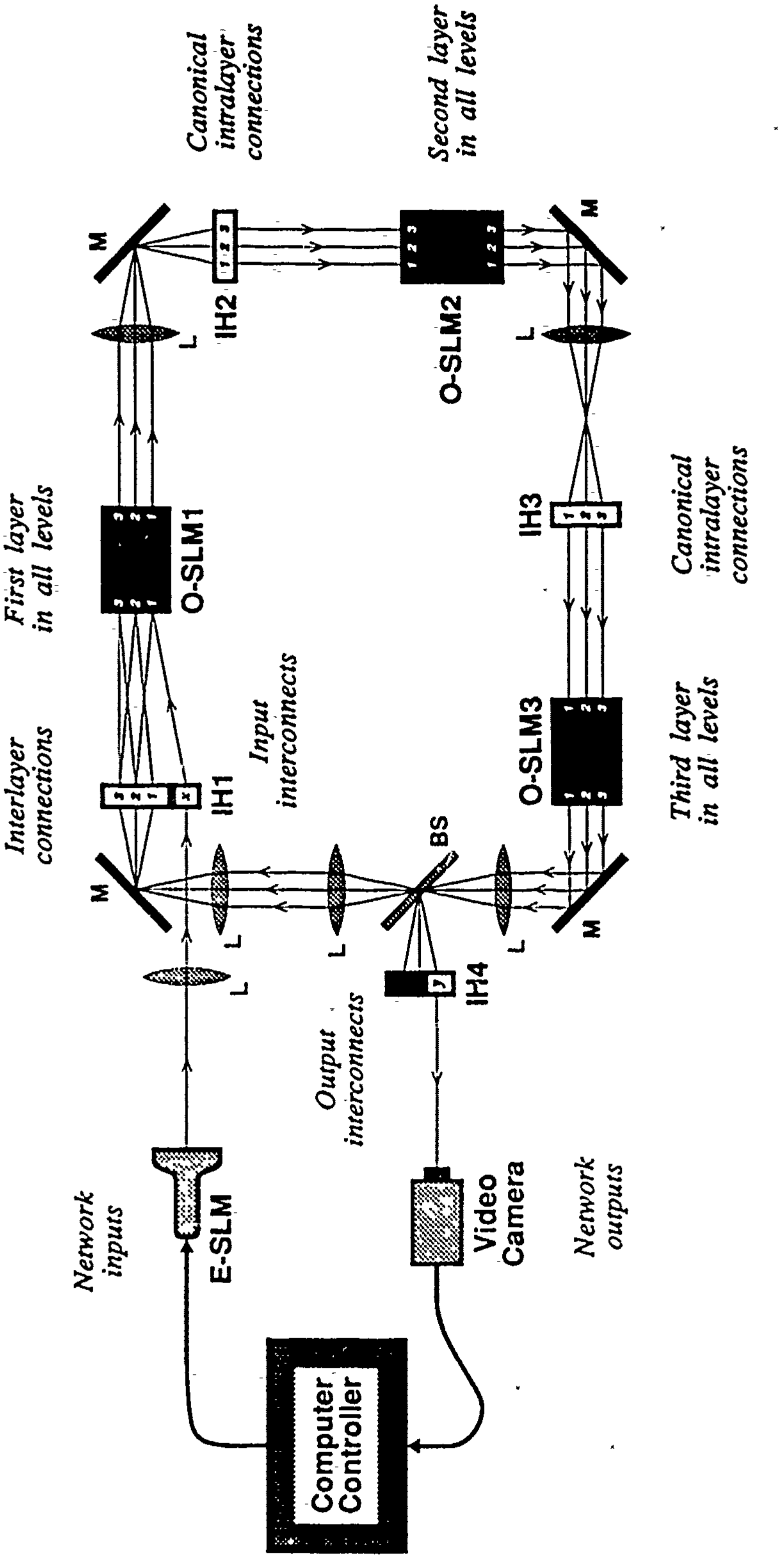


Figure 29: Optical architecture for the SONN shown in Figure 1. The thin lines between the various components are light beams.

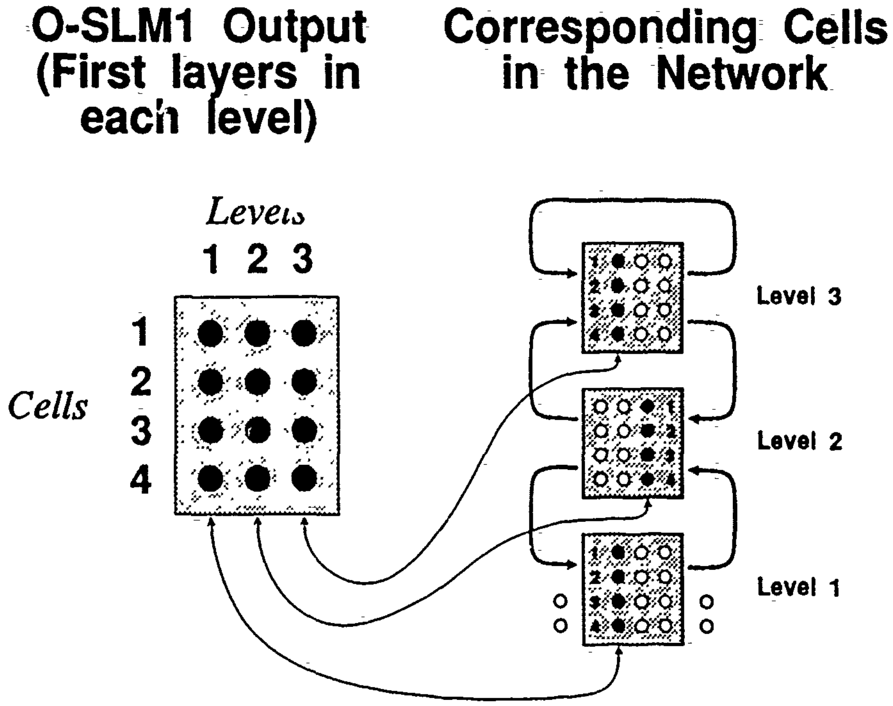

O-SLM1 Output (First layers in each level)
Corresponding Cells in the Network
Levels
1 2 3
Cells
1
2
3
4
Level 3
Level 2
Level 1

Figure 30: Cell layout for O-SLM1 in Figure 29 with respect to the SONN in Figure 1.

realize time delays, one way is to store the most recent N outputs ($y_i[t]$, $y_i[t-1]$, $y_i[t-2]$, ..., $y_i[t-N+1]$) on the O-SLMs and have a shift mechanism update the outputs at each time step. This method only provides integral time delays so no interpolation in time is required (nor is it possible to do conveniently). In addition, the optical devices must have larger space-bandwidth products. For example, if $N = 3$ so the values $y_i[t]$, $y_i[t-1]$, and $y_i[t-2]$ are available, the 4 × 3 cell arrangement shown on the left side of Figure 30 would become the 4 × 9 layout shown in Figure 35. The mappings performed by the interconnect holograms would have to be modified slightly to use the desired (delayed) output.

If the O-SLMs do not have internal shift capabilities, the optical shift technique shown in Figure 36 can be used. This figure depicts an O-SLM from Figure 29 with the shift path. The key aspect of the shift path is the one-pixel offset in the positioning of the left mirror (M) and beamsplitter (BS). This offset causes the O-SLM output $y_i[t]$ to be fed back into the input position corresponding to $y_i[t-1]$. In order to work, this technique requires that the O-SLM output does not change directly with the input. In other words, the input light beams must be detected and measured first and then the output beam updated.

There are two other issues which must be addressed for the optical SONN architecture: implementing mixture of excitation and inhibition with an SLM and realizing finite cell bandwidth. The first issue stems from the fact that the input side of current SLMs are square-law detectors and thus measure only the intensity of the incident light. Since the intensity is always a positive quantity, it is difficult to realize an inhibitory input signal

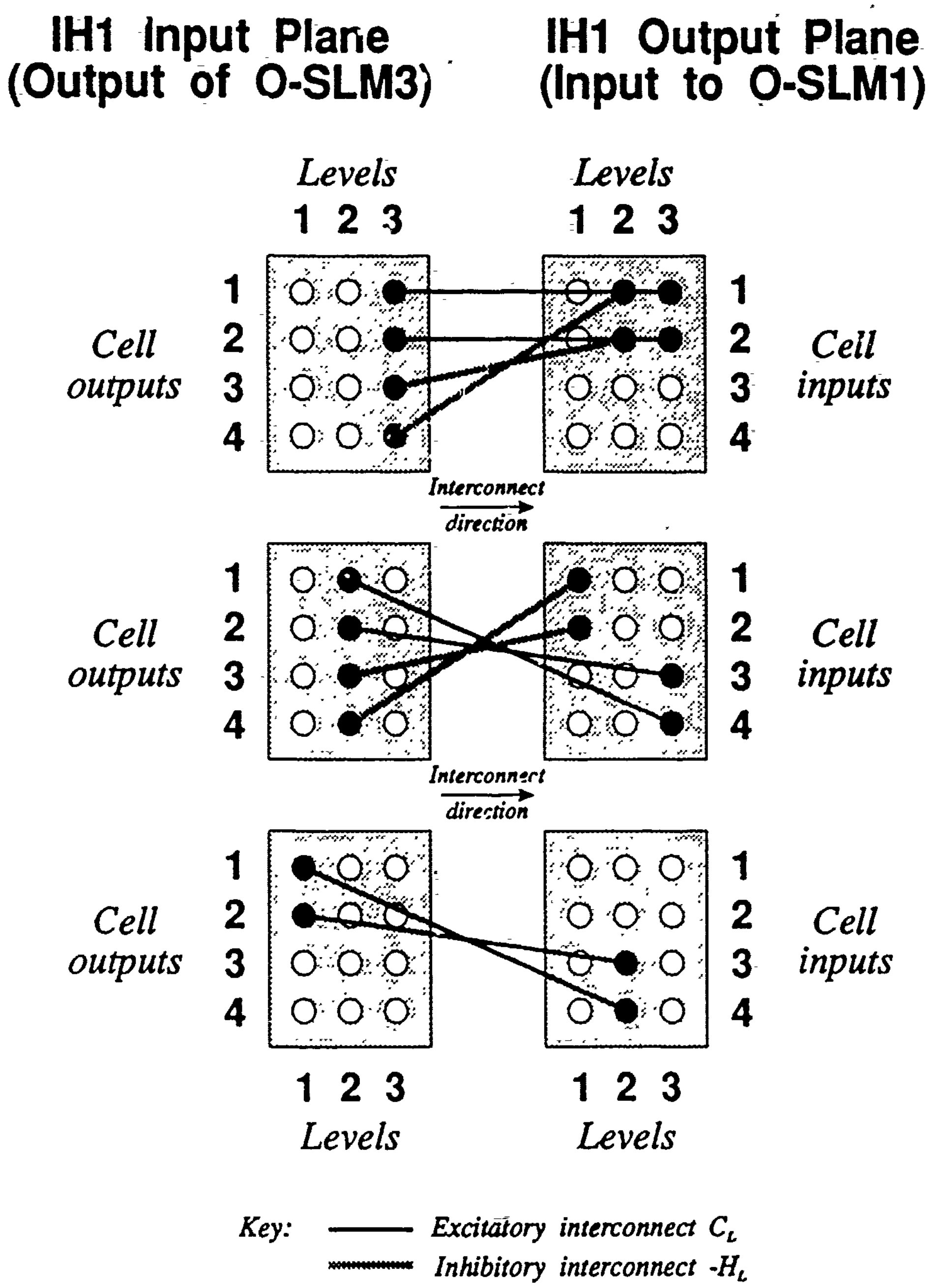


Figure 31: Interlevel connection mapping for IH1 from the cells on O-SLM3 to those on O-SLM1.

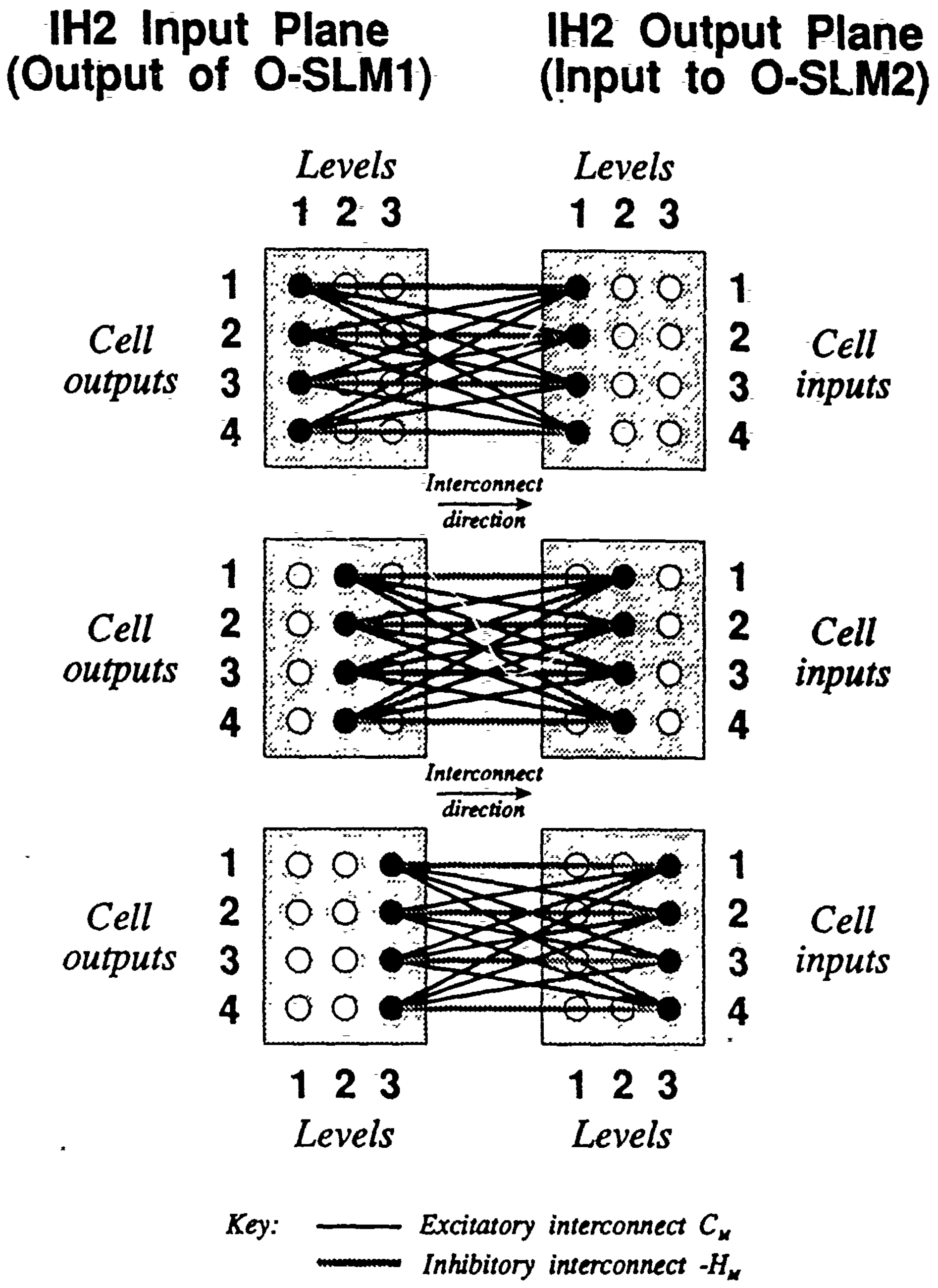


Figure 32: Intralevel connection mapping for IH2 from the cells on O-SLM1 to those on O-SLM2. These interconnects realize the off-center, on-surround canonical topology. The mapping for IH3 from O-SLM2 to O-SLM3 is the same as IH2.

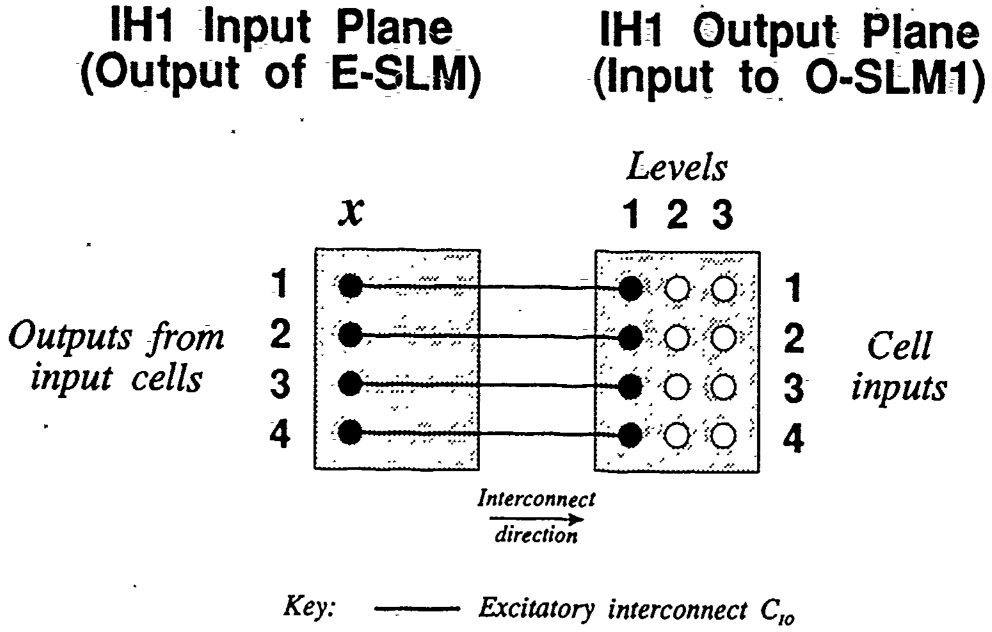


Figure 33: SONN input connection mapping for IH1 from the E-SLM to the cells on O-SLM1.

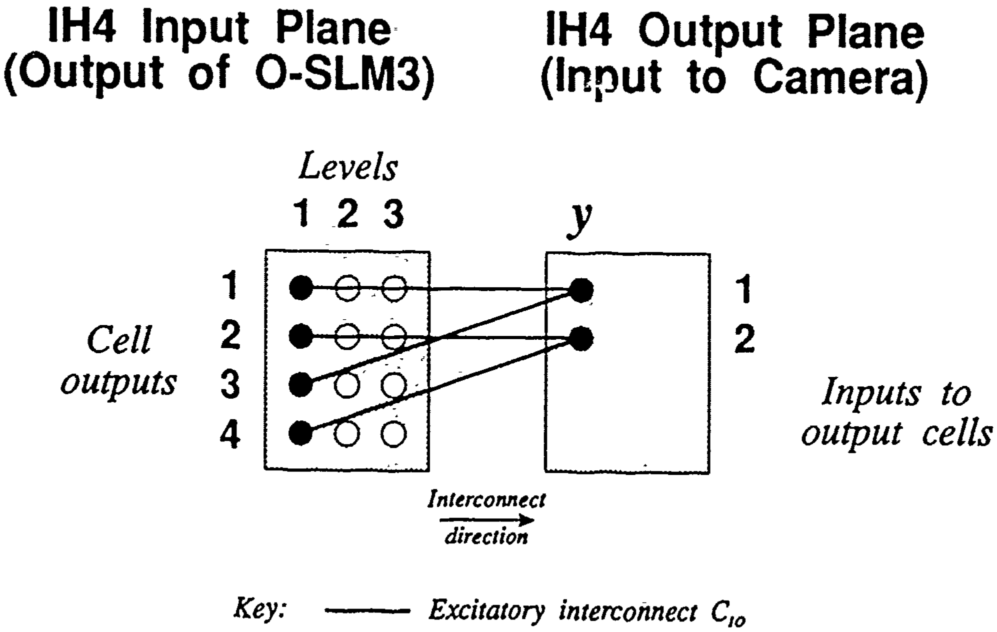


Figure 34: SONN output connection mapping for IH4 from the cells on O-SLM3 to the camera.

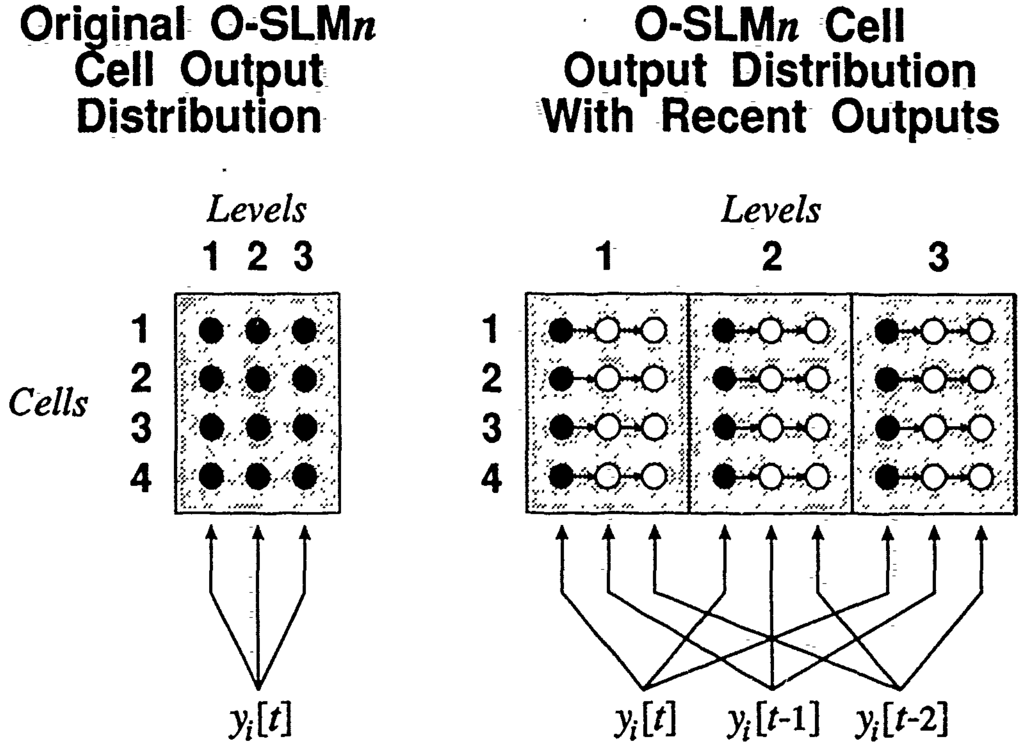

Original O-SLMn Cell Output Distribution
O-SLMn Cell Output Distribution With Recent Outputs
Levels
1 2 3
Cells
1
2
3
4
$y_i[t]$
Levels
1
2
3
1
2
3
4
$y_i[t]$
$y_i[t\text{-}1]$
$y_i[t\text{-}2]$

Figure 35: Cell output distribution on the O-SLMs when recent outputs are retained. The arrows between the cells in the right diagram indicate the shift mechanism at each time step.

to a cell on an SLM when the neighboring cell may have an excitatory input signal. Previous methods such as implementing only inhibitory neurons [3] cannot be used here. The SONN model tries to minimize this problem by having only one inhibitory input to each cell. Unfortunately, this approach does not solve the problem.

The second issue addresses the sum filters in the cells. These filters are necessary for generating piecewise-continuous (i.e., smooth in a discrete-time sense) cycles. The filters are optional if discontinuous limit cycles are acceptable, but this is not the case for the simulated LC-FSM because the TD/IPR networks require continuous limit cycles for the SBP training algorithm to work. Under exposing the SLMs while writing them offers an approximation to the low-pass filter, but only until the SLMs need to be erased to avoid saturation.

With these issues in mind, we have conceived a hybrid optical-VLSI SLM which solves the inhibition and filter problems. In addition, it offers an elegant solution for realizing time delays. A cut-away view of the proposed SLM is shown in Figure 37. Its functional block diagram is shown in Figure 38.

The SONN SLM implements the cell functions electronically and relies on optics to do the interconnections. Unlike previous optical-VLSI devices [4-9], this device relies on three-dimensional integration technique to create an input side and an output side. The input side contains two photodetectors, one for the excitatory light and the other for the

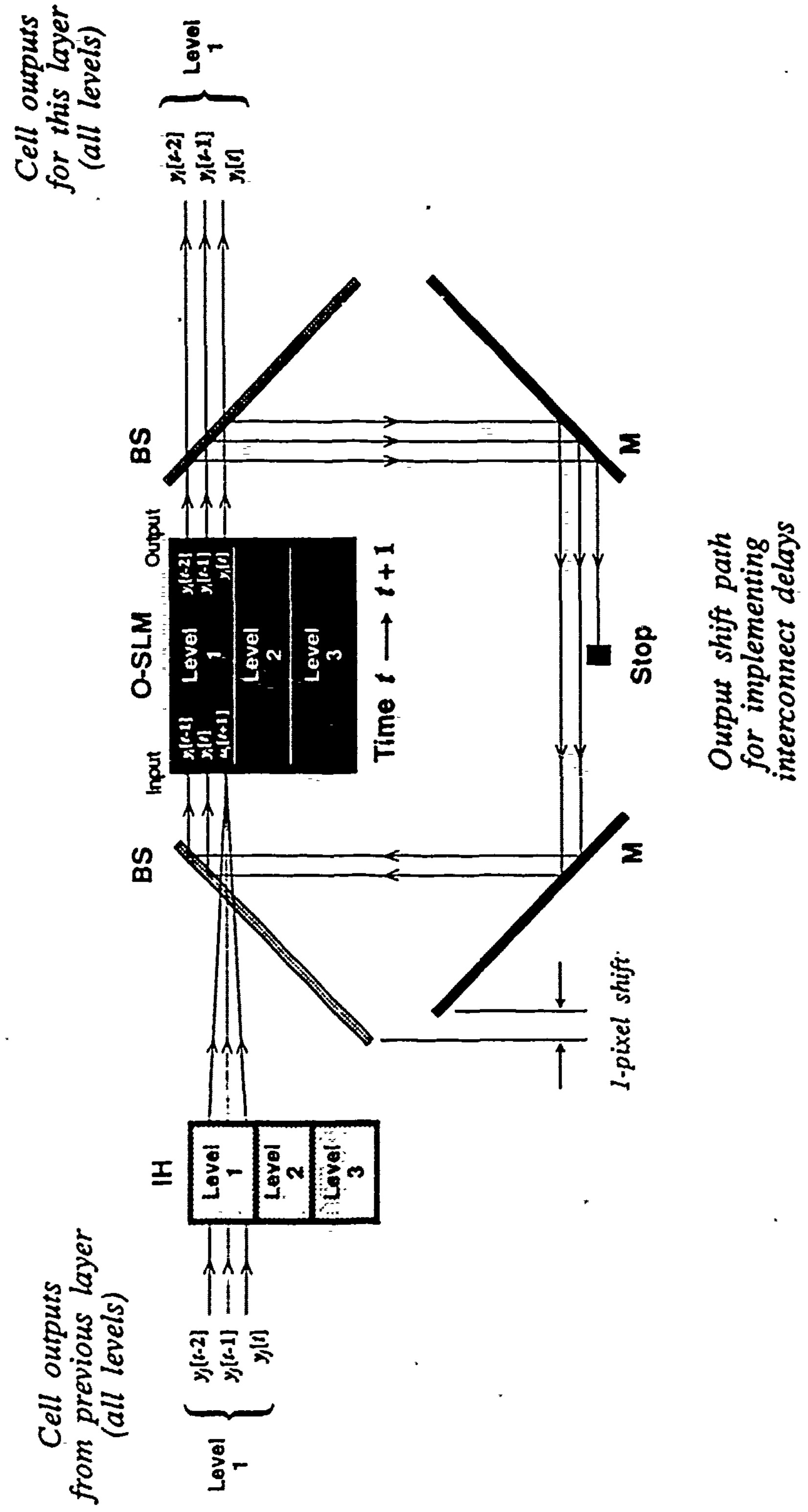


Figure 36: Optical shift technique for saving previous outputs in O-SLMs which do not have internal shift capabilities. Only representative optical paths are shown for clarity.

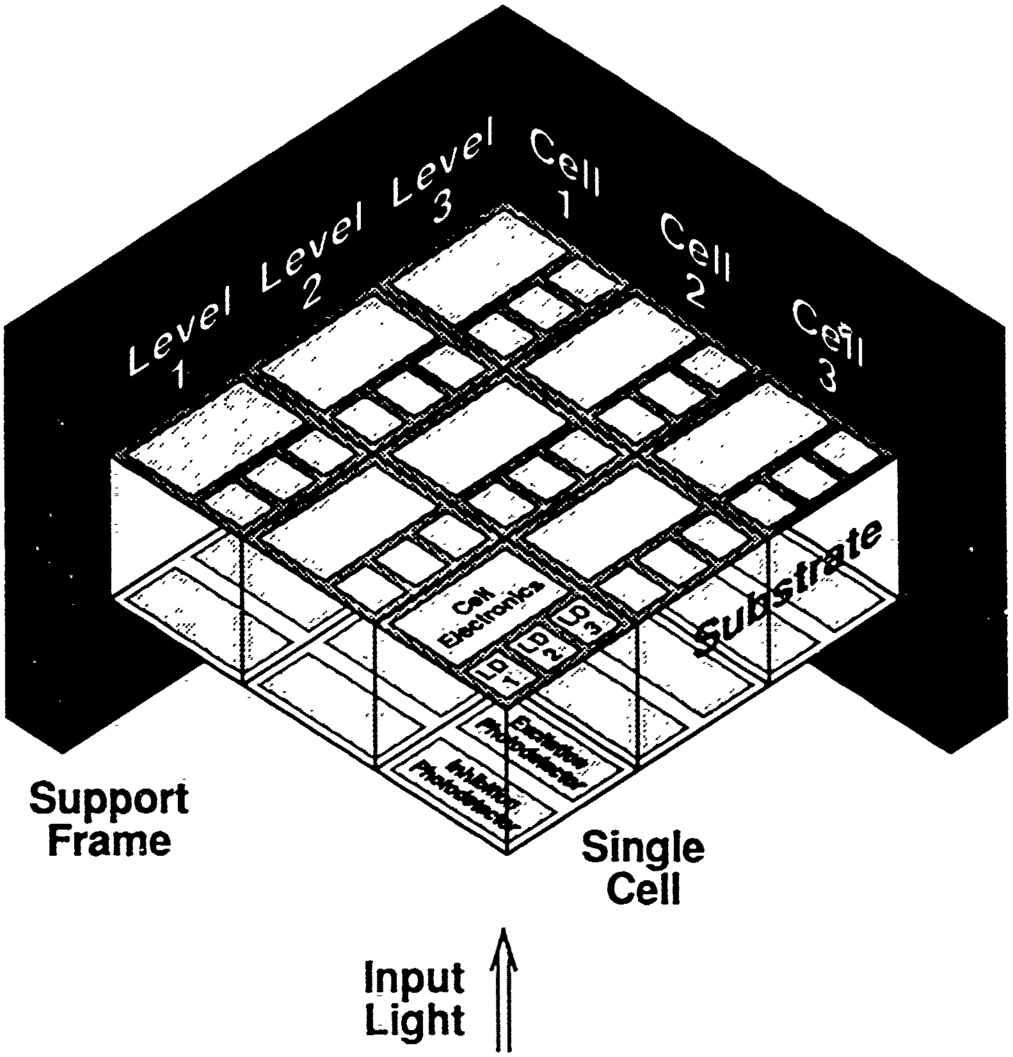


Figure 37: Optical-VLSI SLM for an optical SONN implementation. Each cell has the functional structure shown in Figure 38. Only three of the four cells in a layer as shown in Figure 35 are illustrated here.

inhibitory light. These photodetectors are connected through the bulk substrate of the device via vertically integrated wires to the output side which contains all the processing electronics. Thus, the three-dimensional integration needs to provide only a method for connecting one plane with another plane with a few writes.

A difference amplifier performs the necessary subtraction to compute the total weighted cell input, $u_i[t]$. The cell filter follows and is shown as a simple RC low-pass filter. An electronically controllable switch selects either the filtered form of the weighted sum or the weighted sum itself to be $v_i[t]$. In this way, the filter can be either activated for continuous-time cycles or deactivated for discrete-time cycles.

Finally, $v_i[t]$ is loaded into an analog latch at the next clock pulse. Also with this clock pulse, the outputs of all the latches are loaded into the next respective latch in the shift path. Once the latches have been loaded, their outputs are used to drive individual

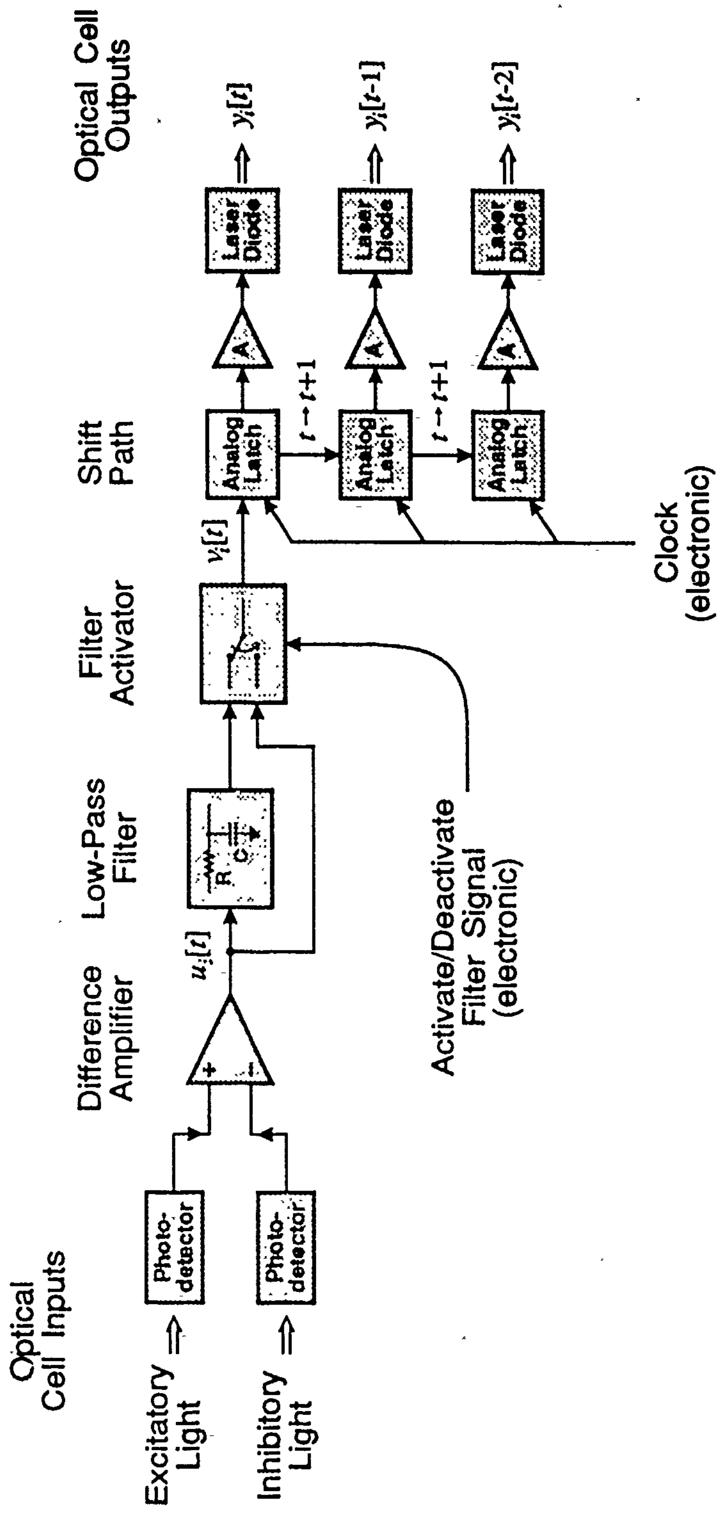


Figure 38: Block diagram of an electronic cell in the optical-VLSI SLM for the optical SONN architecture.

optical light sources such as laser diodes or LEDs via buffer amplifiers. The final output of the cell is a set of optical beams whose intensities are proportional to $y[t]$, $y[t-1]$, $y[t-2]$, ..., $y[t-N+1]$, respectively ($N = 3$ in Figure 38). All beams are on simultaneously. Their uniqueness is maintained by their spatial separation. If time delays are not desired, the shift path can be eliminated with the exception of the first analog latch.

The main disadvantages with the SONN SLM are mechanical support and thermal cooling. Since both the top and bottom surfaces are active, the substrate can be mounted only on its sides. The substrate must be thick enough to provide sufficient rigidity and durability. Fortunately, the thickness of the substrate is not constrained by the electronics because the substrate only contains wires. As for the cooling issue, the cells in the center of the device have limited access to a heat sink. This access is necessary because the laser diodes consume milliwatts of power as opposed to microwatts for all the other electronics.[2]

Admittedly, a SONN SLM with all components mounted on one side would have immediate access to a heat sink and would be easier to fabricate. However, it also would complicate the interconnection scheme because the input and output light paths would conflict. If this conflict could be resolved through a redesign of the interconnection holograms, the one-sided fabrication offers superior mechanical support, thermal cooling properties, and well-developed manufacturing techniques.

5.2 Computer-Generated Hologram Development

In additioa to the SLMs, the interconnection holograms (IHs) form the other major component of the optical SONN architecture. Considering the practical optical demands of the SONN architecture, general, flexible, and efficient IHs were sought. A computer generated approach seemed the best suited for the requirements, but after a review computer generated hologram (CGH) fabrication techniques, we were motivated to develop a CGH process to create IHs possessing high efficiencies, generality, low processing costs, and expedient scheduling. Current high resolution color printer technology was used as a mechanism for creating multiple discrete phase levels in bleach processed silver halide photographic film. This technique allowed for the creation of arbitrary IHs possessing the necessary high efficiencies.

Several techniques for creating CGHs have employed either binary or multiple step representation for amplitude-only, phase-only, or amplitude and phase CGHs. The Lohmann and Lee 1974 binary amplitude-only holograms encode amplitude and phase information by defining a blocking area and its displacement within a CGH cell, but of the many binary amplitude-only encoding techniques [12], these methods possess the highest diffraction efficiencies of 0.1-0.2%. These techniques are easily implemented with

[2] One possible solution is to place a glass covering over all the laser diodes. The thermal conductivity of glass is an order of magnitude larger than that of air and thus would act as a better albeit transparent heat sink. Another possible solution is to make the substrate thick enough to allow a miniature cooling system to be incorporated into it. Alternatively, the laser diodes could be removed entirely from the SONN SLM and could be replaced by a reflective electro-optic material such as a miniature liquid crystal display. In this case, the latches in the shift path in Figure 38 simply drive a high-impedance capacitive load which requires very little power. The SONN SLM then would have to be read out using an external laser with an expanded beam.

a laser printer and photoreduction system but result with very poor optical efficiencies. Multiple step amplitude-only CGHs encode phase and amplitude information by discrete approximating the continuous gray level spectrum found when holograms are recorded optically, but this technique requires a specialized gray level photoplotter and yields marginal efficiency considering the amplitude-only nature.

A natural extension for efficiency improvement is phase-only encoding to eliminate absorptive loss. Binary phase approaches include binary Dammann gratings [13] which provide controlled amplitude and phase relations in 1-D or uncoupled 2-D or direct gratings for Fresnel lens implementation [14]. Dammann gratings provide efficiencies up to about 75% but require very accurate phase transition placement and are impractical for coupled 2-D functions. Binary Fresnel lens implementations are limited to a maximum 41% efficiency. To improve these efficiencies, multiple phase level approaches including the kinoform and blazed surface relief elements have been considered [15, 16]. These techniques represent the highest efficiencies achievable, but either require gray level photoplotting or capital intensive processing equipment for glass etching.

Each of the above processes have their own merit, but high efficiency, low cost, flexible, and expedient processed CGHs were required for the SONN architecture. Our technique creates multiple phase levels in bleach processed black and white photographic film by modulating film exposure with colors possessing differing spectral transmissions. Figure 39 illustrates the entire process with cartoon style flow diagram. A convention IBM compatible personal computer was used, but any computer is possible provided the PostScript, a powerful illustration/documentation printer programming language, is the ultimate file format. The PostScript program contains all necessary commands to generate the required color placement for CGH and IH implementation. The resultant color mask was photoreduced with standard high resolution camera equipment on high resolution black and white film, and photochemical processing developed and stabilized the phase-only CGH image.

Current technology has allowed for the availablity of inexpensive color printers that offer high resolution output. Of the many possible choices, we selected the QMS ColorScript 100 Model 10 for the quality of output and good price/performance ratio. This printer cost ~ $8,200 and offered 300 dots per inch resolution with a total of eight solid colors. Output from the printer was taken on clear transparency film for backlite illumination during the photoreduction process. We constructed a vibration resistant high resolution backlite copy stand for 30X photoreduction with off-the-shelf high quality optical components (camera, lens, and white sources). The input image plane possessed 5 to 10% illumination uniformity, and the photoreduced results were capable of approximately 200 line pairs per mm at an optical density of 3.0.

The format of the camera was 4×5 inches, and we used high resolution black and white film of that dimension. The selection of the film was extremely important for the distribution of color exposed/recorded optical densities (in turn, optical phases). Each color acts as a spectral transmission filter to the backlite white light source, and thus narrowing the spectral power density impinging onto the black and white film. The monochrome equivalent (luminance) of the printer colors yields a relatively linearly stepped growing relation [17]. To maintain this relation, a film with a relatively flat spectral sensitivity was necessary so that linearily distributed optical densities would

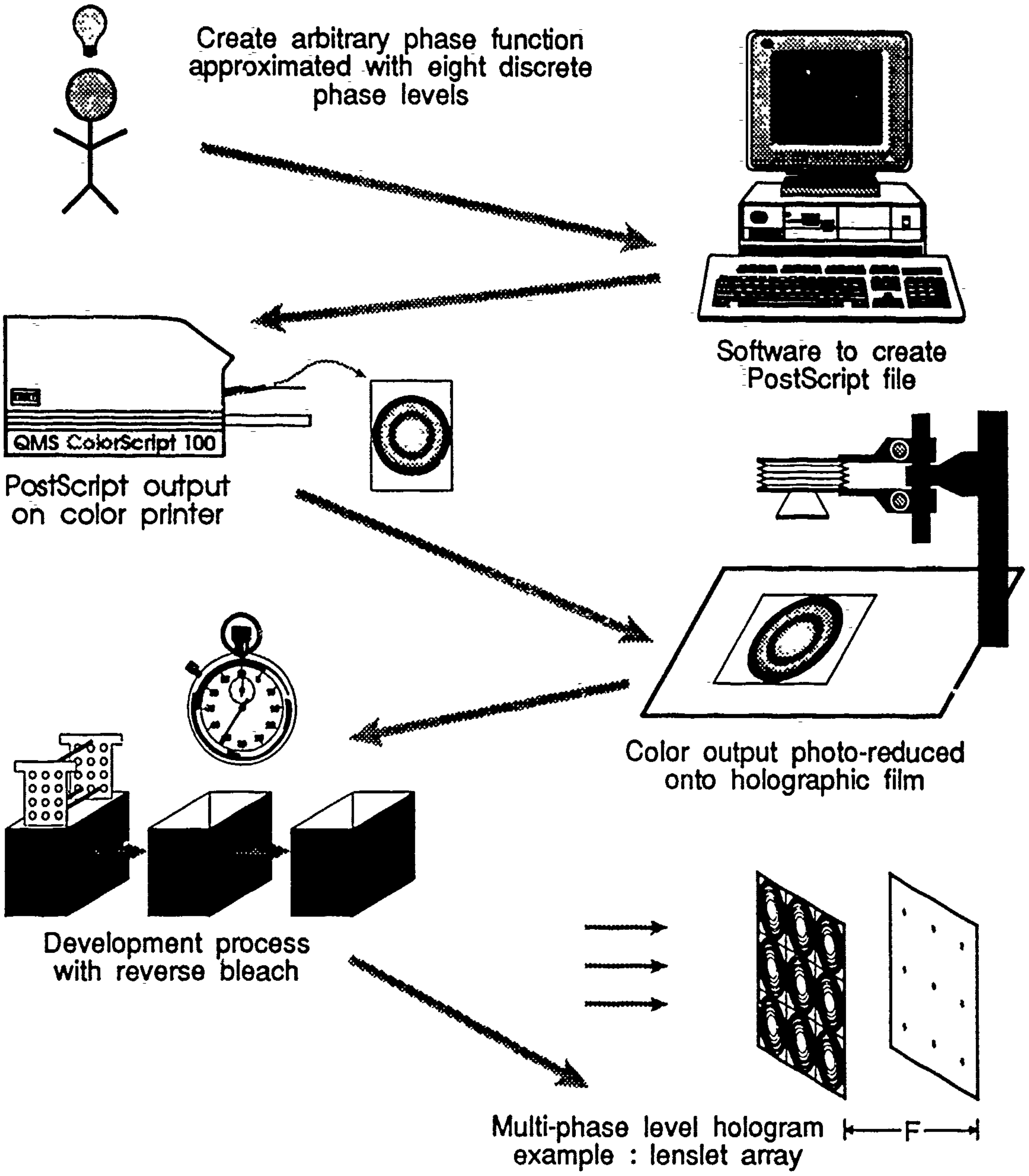

Create arbitrary phase function approximated with eight discrete phase levels
Software to create PostScript file
QMS ColorScript 100
PostScript output on color printer
Color output photo-reduced onto holographic film
Development process with reverse bleach
Multi-phase level hologram example : lenslet array
F

Figure 39: Flow diagram of the color CGH process.

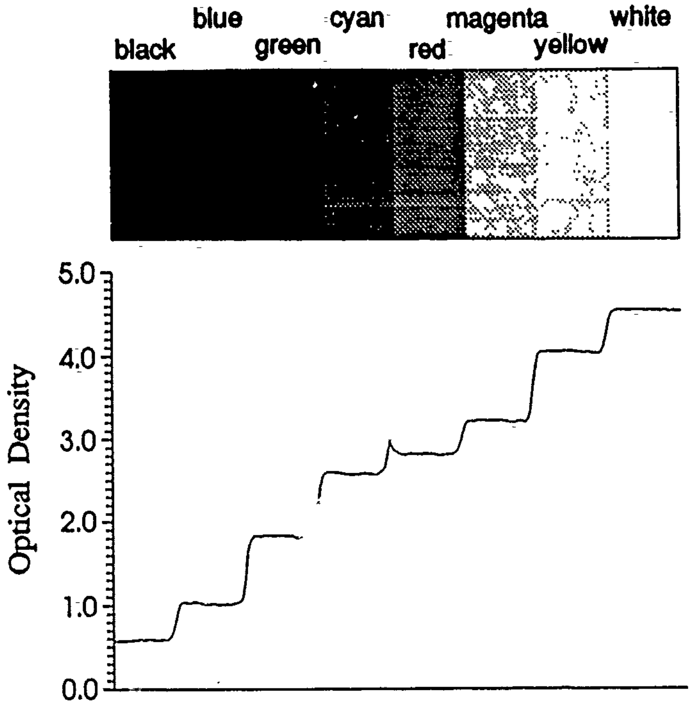


Figure 40: Nominal exposure color to density mapping for Kodak 649F.

result. Kodak 649F on a 4×5 inch glass substrate was selected, and the emperical mapping of color to optical density is illustrated in Figure 40. For Kodak 649F, we observed repeatable mapping for film plates within the same emulsion batch given uniform processing conditions, but differing emulsions exhibited different photographic speeds requiring exposure characterization for each emulsion. The chemical processing times and conditions were optimized for the desired color mapping as illustrated [17].

Discrete phase levels were created by modifying the developing process to include bleach. A reversal bleach was selected for removing the pigment of the latent image by eliminating all of the exposed grains. This process yields less phase, or retarded phase, in the emulsion regions that experienced higher net exposures. The emperical color to phase mapping is illustrated in Figure 41. We measured the phase changes with a modified film substrate grading technique we developed. A slight angle wedge (imperfect microscope slide) was sandwiched to the film plate being measured with index matching fluid, and this wedge provided closely spaced, approximately straight fringes when the sandwich was read out in reflection with a colimated laser source. The discontinunity of the fringes due to differently (color) exposed regions was measured. By organizing the colors by growing luminances, the ambiguity in phase measurement was eliminated.

The major limiting factor for our color to phase mapping process was the photoreduced rolloff affected by printer anomalies and camera diffraction limitations. We observed poor print quality at resolutions (one to three printer pixels per color) near the resolution limit of the printer we selected. The modulation transfer function (MTF) for 30X reduced color gratings was measured for optical density and illustrated in Figure 42, and by optical density to phase correlation, the phase rolls off in the same fashion. This

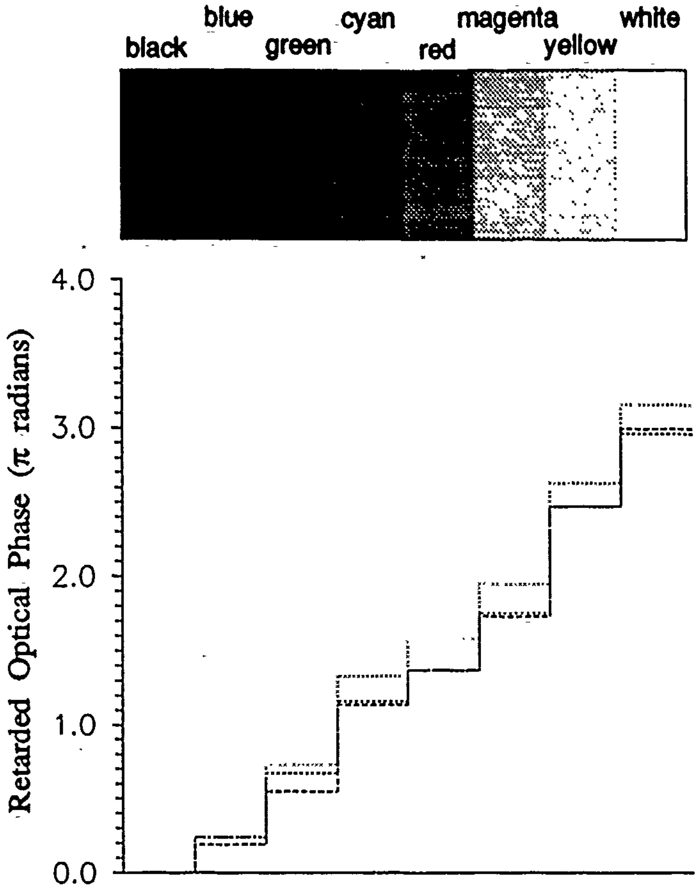


Figure 41: Color to phase mapping at nominal exposure for several trial plates of Kodak 649F.

rolloff issue was observed to affect the diffraction efficiency of the example applications.

As the first application, eight phase level approximated Fresnel lens were implemented. The Fresnel lens diffraction efficiency was modelled for eight phase levels with respect to the non-uniform phase step height and/or widths [17]. A maximum diffraction efficiency of ~ 86% was determined (excluding insertion losses) for a given relative density distribution of eight colors with uniform spacing. A sample side on view (optical density - reverse contrast) of an example Fresnel lens implemented with color mapping is shown in Figure 43, and bleaching removes the reversed contrast. This figure also illustrates how the outer Fresnel zones (smaller feature sizes) were rolled off which resulted in a non-uniform phase modulation depth over the extent of the lens effectively reducing diffraction efficiency.

We constructed a variety of single and compound Fresnel lenslet arrays possessing differing focal lengths and aperture sizes. An example of two lenslet arrays is illustrated in Figure 44. We observed a peak measured efficiency of 67.2% for eight evenly spaced colors (excluding insertion loss). Rolloff into the outer Fresnel zones retarded the maximum diffraction efficiency from the 86% predicted peak. The highest efficiency 30 cm lens exhibited a 130 μm spot size corresponding to a Gaussian beam predicted size of 117.4 μm given the lens focal length and diameter of 2.4 mm. An insertion loss of 26.3%

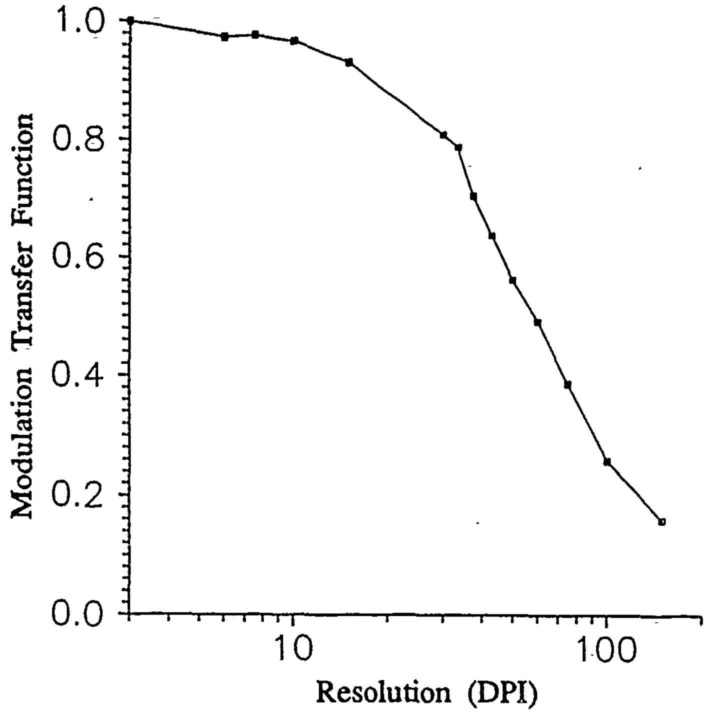


Figure 42: Modulation transfer function of the 30X photoreduced printer colors as the feature size decreases (in printer resolution units).

due to front and back surface Fresnel reflection, scattering, and a high absorption losses was experimentally measured. A large absorption loss was due to staining by the bleach used, but other reversal bleaches may possess lower insertion loss attributes. By exposing lenses within an array differently, we were able to produce lenses of varying diffraction efficiencies which allowed us to quickly characterize different film emulsions for maximum diffraction efficiency with only one film plate.

Ultimately, we implemented optical interconnections that offered a completely arbitrary nature with high optical power efficiency. Arbitrary interconnections can be created with discrete phase level approximated diffractive blaze gratings (synthetic blazed gratings) via a computer and this process. Figure 15 illustrates how the grating were constructed with color to phase mapping. The blazed grating was used in transmission, and off-axis diversion or elevation, θ, was specifing with the period of the grating. Phase depth of the grating is set to 2π or one wavelength, λ, and thus the off axis diversion is specified by

$$\tan\theta = \frac{\lambda}{\Lambda}$$

where Λ is the period of the grating. For the interconnection examples we addressed, the vector relative, x and y, displacements from the input plane to the output plane defines the grating period as follows

$$\Lambda = \frac{\lambda d}{\sqrt{x^2 + y^2}}$$

where d is the throw distance between the input plane and the output plane. The grating

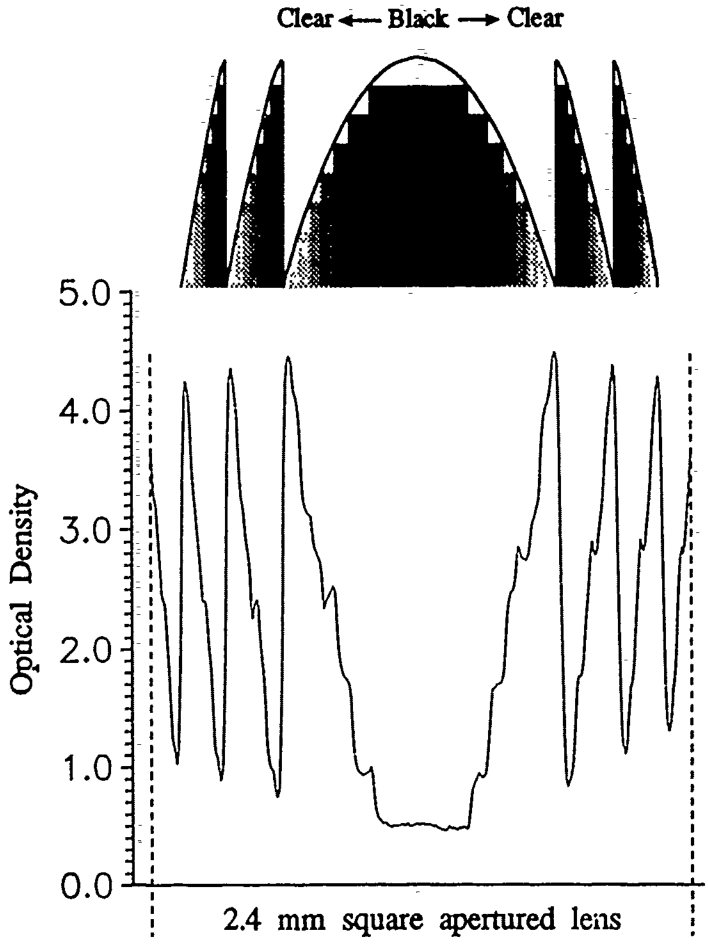


Figure 43: Optical density profile of a 30 cm Fresnel lens.

lines are rotated by

$$\phi = \tan^{-1} \frac{x}{y}$$

within the input connection cell to provide azimuth control.

With .his interconnection methodology, we implemented an arbitrary interconnection with a 5×5 input plane and spelled out "MIT" in the output plane. Off-axis and on-axis cases were considered with emphasis placed on the on-axis case. The input-output mapping for the on-axis case is illustrated in Figure 46. Using PostScript, the color mask(s) necessary for the IH was generated and photoreduced with an exposure level to optimize diffraction efficiency. Figure 47 illustrates a photograph of the output plane, and threshold functionality of photographic film effectively eliminated anomalous on or off-axis noise.

For the on-axis case, the average diffraction efficiency was 54.2% with a peak efficiency of 86.0% at the lower right corner of the M. The average contrast ratio was 11.5:1 with a best case value of 18.2:1 and worst case value of 7.2:1 calculated against the average noise power. The off-axis MIT interconnection exhibited similar efficiency, but it possessed a

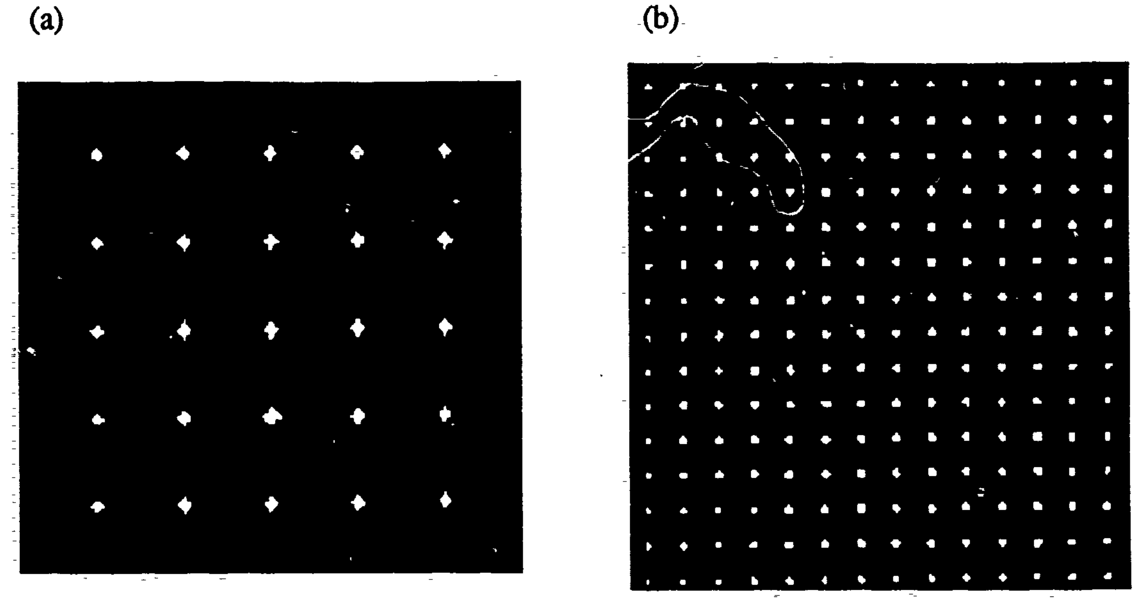


Figure 44: Focal plane photograph of (a) a 5×5 30 cm lenslet array and (b) a 15×15 5 cm lenslet array.

higher average contrast ratio of 17.4:1 with a 22.6:1 best case and 13.6:1 worst case because the interconnection was directed out of the on-axis power region. Given the rolloff and diffraction efficiency as a function of resolution, off-axis deflection, θ, of 0.6° with at least 50% efficiency was realizable which corresponds to a 7.5 μm minimum feature size. Deflection of up to 0.78° with reduced efficiency was possible before printer problems become insurmountable.

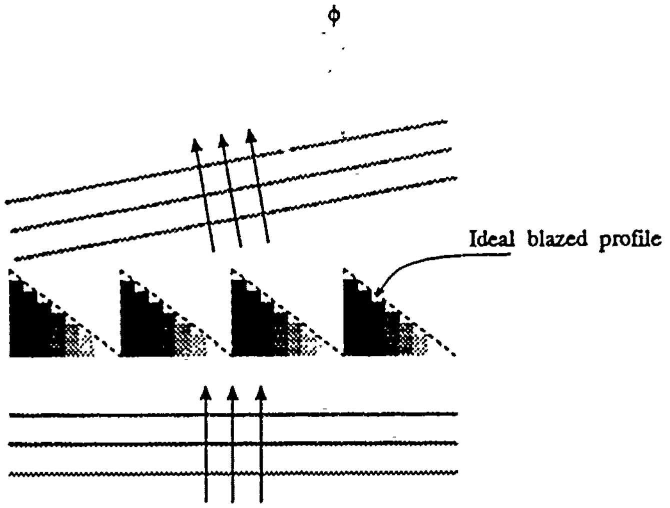


Figure 45: Construction of a synthetic blazed grating with the color to phase mapping CGH process.

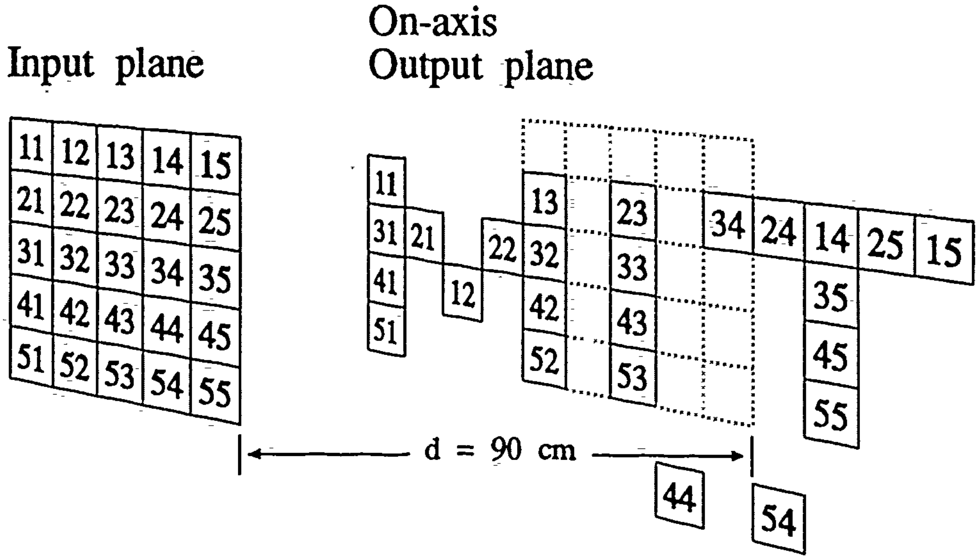


Figure 46: On-axis MIT interconnection mapping.

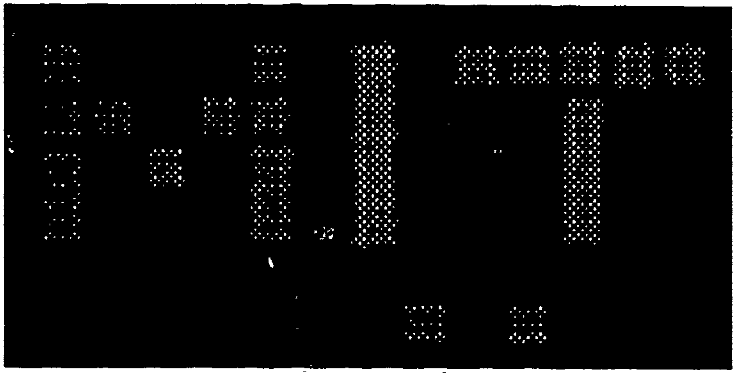


Figure 47: Experimental construction of an on-axis MIT interconnection.

Considering these numbers, this color CGH process has provided an excellent mechanism for producing on-axis IHs with efficiencies > 50% and good contrast ratios. Other interconnection examples including the optical folded perfect shuffle and the sub-element interconnection array for arbitrary fan-out (9 total connections) of input cells were implemented with similar successes [17].

5.3 Computer-Controlled SLM Characterization

We also developed a computer controlled system to quickly evaluate the limitations of spatial light modulators (SLMs) used within our optical systems. The parameters controlled and measured included framing rates, framing dynamics, contrast ratios, MTFs, optical sensitivities, exposure dynamics, output uniformity, and input imaging. A PC controlled data acquistion system with digital to analog output was setup with serial controlled linear translation and rotation stages. Optical powers were measured with a high sensitivity autoscaled power meter interfaced to the central PC with an IEEE-488 connection.

With a computer controlled Michelson interferometer, we were able to create arbitrary high spatial frequency fringes and record them onto a microchannel spatial light modulator (MSLM), an O-SLM, under test. Using these fringes, we mechanically scanned the magnified MSLM output image with a fine razor blade apertured high gain photodetector, and under computer control, we collected all the necessary data to compile a MTF curve for the device under test. By recording a uniform input exposure, we scanned the x and y output plane with pinhole mounted to the high sensitivity optical power meter detector to determine the device output uniformity. The control software was written in a fashion to maximize flexiblity and reconfigurability dependant upon application.

6 Conclusions

The original goal of the program was to develop hybrid optical inference machines. Instead of expanding upon the conventional approaches (based on nonlinear matrix-vector multipliers), we opted to look at a different method. Because of problems with fault tolerance in the conventional approaches, we chose to consider encoding information using limit cycles. This new approach created a whole new set of challenges and problems to overcome. During the program, we solved many of the problems associated with processing with limit cycles, but some issues remain unsolved with respect to a practical optical implementation.

Given the unfamiliarity of processing with limit cycles, we chose to concentrate on a simplified form of symbolic processing, the finite state machine (FSM). The limit cycle form of this computation paradigm (LC-FSM) requires (1) a medium that supports many cycles and (2) a method for establishing couplings between cycles. The first task corresponds to an associative memory for limit cycles and the second one is a controller for switching between cycles. Because of their flexibility, neural networks were chosen as the working medium.

In the program, we created the self-oscillating neural network (SONN) model for solving the first task. This model was designed with an optical implementation in mind. It is very tolerant of static variations in the network parameters (easily in excess of ±20%). In an optical implementation, these variations appear as nonuniformities in the spatial light modulators (SLMs), the readout light, and crosstalk in the holographic interconnections. The SONN is well suited to the first task because it has many cycles available with no training or programming required. They can be selected by either constant or cyclical inputs. The SONN is an example of a component which will work as an associative memory for limit cycles in a LC-FSM.

For the second task, we derived the spectral back-propagation (SBP) training algorithm for creating a LC-FSM controller for limit cycles. This algorithm is an extension of the conventional back-propagation algorithm to train input-output sequences. The SBP algorithm uses discrepancies in the Fourier series spectra of the output sequences as an error criterion. This approach allows not only the weights but also the time delays associated with the interconnects to be trained. Furthermore, the cells in the network can have finite bandwidth via a first-order low-pass filter. We have demonstrated the algorithm successfully on both feedforward and recurrent networks with both continuous-time and discrete-time sequences.

In a simulated 3-state LC-FSM with 8 possible transitions, the SBP algorithm allowed us to develop a very simple set of networks (the transition detector and the in-phase recognizer, TD/IPR) for recognizing a particular multidimensional limit cycle. These networks can do the recognition in under 3 periods with a very high amount of phase sensitivity (7°). They permit the same input and current-state cycle of the LC-FSM to stimulate different transitions depending upon their relative phase. Furthermore, the portions of the TD/IPRs whose weights and time delays are trained by the SBP algorithm are linear networks with no hidden cells. Thus, the TD/IPRs can be trained very quickly. These networks form the key component of the LC-FSM controller. They generate binary signals, each of which corresponds to a unique transition condition in the LC-FSM. The transitions then can be made using a simple mapping network. The SBP algorithm thus is useful tool for creating a LC-FSM. Using the SONN and the SBP algorithm, we were able to demonstrate successfully a working LC-FSM using computer simulations.

The problems left unanswered by the program are associated with creating a practical optical LC-FSM, a intermediate but necessary milestone towards making a working hybrid optical inference machine. We designed an optical architecture for the SONN and developed a novel technique for making the required interconnection holograms using conventional color printer technology. This IH foundry provided efficiencies exceeding 50% expediently, flexibly, and at a low cost. We demostrated arbitrarily defined IHs with the color CGH process. However, limitations with current SLMs motivated us to propose a new general-purpose hybrid optical-VLSI SLM. Development of this type of device would greatly benefit the research into hybrid optical inference machines, particularly those based on limit cycles.

7 References

1. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Vol. 1* (MIT Press, Cambridge, MA, 1986).

2. T. Tollenaere, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties," Neur. Net. **3**, 561 (1990).

3. W. Kawakami, H. Yoshinaga, and K. Kitayama, "Demonstration of an Optical Inhibitory Neural Network," Opt. Lett. **14**, 984 (1989).

4. E. A. Rietman, R. C. Frye, C. C. Wong, and C. D. Kornfeld, "Amorphous Silicon Photoconductive Arrays for Artificial Neural Networks," Appl. Opt. **28**, 3474–3478 (1989).

5. R. I. MacDonald and S. S. Lee, "Photodetector Sensitivity Control for Weight Setting in Optoelectronic Neural Networks," Appl. Opt. **30**, 176–179 (1991).

6. E. A. Rietman, R. C. Frye, and C. C. Wong, "Signal Prediction by an Optically Controlled Neural Network," Appl. Opt. **30**, 950–957 (1991).

7. J. Ohta, M. Takahashi, Y. Nitta, S. Tai, K. Mitsunaga, and K. Kyuma, "GaAs/AlGaAs Optical Synaptic Interconnection Device for Neural Networks," Opt. Lett. **14**, 844–846 (1989).

8. Y. Nitta, J. Ohta, K. Mitsunaga, S. Tai, and K. Kyuma, "Optoelectronic Associative Memory Using an Advanced Optical Neurochip," Appl. Opt. **30**, 1328–1330 (1991).

9. M. Govindarajan and S. R. Forrest, "Optically Powered Arrays for Optoelectronic Interconnect Networks," Appl. Opt. **30**, 1335–1346 (1991).

10. C. Warde and J. A. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Appl. Opt. 26, 940 (1986).

11. J. A. Kottas, "Trends in Knowledge Base Processing Using Optical Techniques," *Proc., 1989 IEEE Inter. Conf. Sys., Man, & Cyber.* (Cambridge, MA, November 1989), p. 1250.

12. G. Tricoles, "Computer generated holography: a historical review," Appl. Opt. **26**, pp. 4351-4360 (1987).

13. H. Dammann, K. Görtler, "High efficiency in-line multiple imaging by means of multiple phase holograms," Opt. Comm. **3**, pp. 312-305 (1971).

14. K. Rastani, A. Marrakchi, S. F. Habiby, W. M. Hubbard, H. Gilchrist, and R. E. Nahory, "Binary phase Fresnel lenses for generation of two-dimensional beam arrays," Appl. Opt. 30, pp. 1347-1354 (1991).

15. L. B. Lesem, P. M. Hirsch, and J. A. Jordan, "The kinoform: a new wavefront reconstruction device," IBM Journal of R&D **13**, pp. 150-155 (1969).

16. G. L. Swanson, "Binary optics technology: the theory and design of multi-level diffractive optical elements," MIT Lincoln Lab. Tech. Rep. **854**, (1989).

17. V. E. Shrauger, "Development and applications of computer generated phase only optical interconnection elements," S.M. Thesis, MIT, Cambridge, MA (September 1991).

8 Hybrid Optical Inference Machines: Architectural Considerations [10]

Reprinted from Applied Optics, Vol. 25, Page 940, March 15, 1986

Hybrid optical inference machines: architectural considerations

Cardinal Warde and James Kottas

A class of optical computing systems is introduced for solving symbolic logic problems that are characterized by a set of data objects and a set of relationships describing the data objects. The data objects and relationships are arranged into sets of facts and rules to form a knowledge base. The solutions to symbolic logic problems involve inferring conclusions to queries by applying logical inference to the facts and rules. The general structure of an inference machine is discussed in terms of rule-driven and query-driven control flows. As examples of a query-driven inference machine, two hybrid optical system architectures are presented which use matched-filter and mapped-template logic, respectively.

I. Introduction

A. Definitions

Symbolic logic problems involve, in an abstract sense, a set of data objects and a set of relationships describing the data objects. The data objects and relationships constitute a knowledge base which is generally arranged as sets of facts and rules. A fact is a statement connecting a relationship with one or more data objects so that the statement is always interpreted as true. On the other hand, a rule is a statement which defines a relationship using other relationships, data objects, and/or facts.

A symbolic logic problem is usually stated in the form of one or more queries which are questions concerning relationships and data objects. The queries are answered by applying logical inference to the knowledge base of rules and facts. This inference process generates a set of assertions (inferred facts) from the knowledge base. The solution to the queries, therefore, becomes a set of conclusions in the form of data objects, which is inferred from the set of assertions so as to satisfy the queries.

B. PROLOG

Symbolic logic problems are relatively common. They arise in areas such as expert systems and other artificial intelligence applications. In recent years, the computer science language PROLOG has become a tool for solving these types of problem on electronic computers.[1] For example, two goals of fifth-generation computers are (1) to develop a machine capable of logical inference and data base operations and (2) to design a language based on PROLOG that would be suitable for inferring and representing knowledge.[2]

To solve a query, electronic PROLOG sequentially searches for the knowledge base for the appropriate rules and facts. This search process uses a flexible pattern-matching technique called unification which involves searching, matching, and backtracking through the knowledge base.[3,4] The performance of electronic PROLOG is limited by its use of serial searching and backtracking. PARALOG, an implementation of PROLOG which uses parallel unification, addresses this issue and is currently under investigation.[2]

C. Role of Optics

It is well known that 2-D parallel optical processors inherently perform high-speed pattern matching. Such systems should, therefore, be more efficient at searching than their serial electronic counterparts because the parallelism eliminates the need for backtracking through the knowledge base. Furthermore, since searching and pattern matching processors do not require high accuracy or large dynamic range, optical processors should in principle be well suited for symbolic logic processing.

We believe, however, that optical inference machines should be designed to be compatible with electronic computers. The goal should be to exploit the strengths of both systems so as to realize hybrid inference machines that are more efficient and versatile than either purely electronic or optical computers. For example, an optical inference machine could potentially be integrated into an electronic fifth-generation computer so that a hybrid machine capable of operating at speeds in excess of 10^9 logical inferences per second (LIPS) could be produced.

The authors are with MIT Department of Electrical Engineering & Computer Science, Cambridge, Massachusetts 02139.

Received 19 June 1985.
0003-6935/86/060940-08$02.00/0.

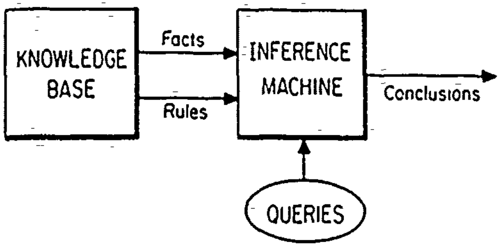


Fig. 1. General structure of an inference machine.

D. History

Previous work in optical symbolic processing was performed by several researchers in the late 1960s and early 1970s. Gabor,[5] Akahori and Sakurai,[6] Nakajima *et. al.*,[7] and Lohmann and Werlich[8] used holography as the basis for their processing techniques. Willshaw *et. al.*,[9] Willshaw and Longuet-Higgins,[10] and Gabor[5] approached the problem using associative network concepts. However, during the 1970s and early 1980s, the emphasis of research on optical computing systems shifted to numerical problems such as matrix–matrix multiplication,[11-13] array processing,[14] and solving sets of linear equations.[15]

More recently, there has been a resurgence of interest in the area of optical symbolic processing. Huang[16,17] has addressed the symbolic problem in a general sense, investigating algorithms and architectures for performing symbolic substitution optically in classical finite-state machines. Furthermore, Huang[18] and Fisher *et al.*[19] have recognized that there may be a possible role for optics in symbolic processors, particularly in solving certain classes of artificial intelligence problems. However, specific applications of optical computers to symbolic logic processing appear, until now, to have been unaddressed.

In this paper, the concepts associated with symbolic logic processors are introduced, and the general architecture of an optical machine capable of inferring logical conclusions from a set of facts and rules is discussed. The general system is approached from two different information flow patterns: rule-driven and query-driven flow. Two hybrid optical realizations for a query-driven inference machine are presented which use classical matched-filter logic and mapped-template logic, respectively. The intent here is to describe these systems from a conceptual point of view. Therefore, no attempt is made to address all the issues involved in realizing a practical system.

II. General Inference Machine Architecture

The general structure of an inference machine is shown in Fig. 1. It accepts as input a set of facts and a set of rules from the knowledge base and one or more queries. The output of the inference machine is a set of specific conclusions which are logically inferred from the facts and rules in response to the queries.

For example, a set of data objects could be a set of names of people. For illustrative purposes, let this set be denoted as

$$\mathbf{D} = \{\text{Karen, Beth, Peg, Liz, Sue, Jean, Ruth, Mike, Tom, Bill, Jim, Fred, Bob, Sam}\}. \tag{1}$$

A set of relationships for **D** might be the possible relationships between the people, such as marriage, mother, father, male, and female. Let this set of relationships be denoted by

$$\mathbf{R} = \{\text{married-to, mother-of, father-of, son-of, daughter-of, child-of, is-male, is-female}\}. \tag{2}$$

The data objects and relationships are linked as a collection of facts and rules which relate the elements of **D** and **R**. In this example, the facts could be defined as

Mike is-male.
Tom is-male.
Bill is-male.
Jim is-male.
Fred is-male.
Bob is-male.
Sam is-male.

Karen is-female.
Beth is-female.
Peg is-female.
Liz is-female.
Sue is-female.
Jean is-female.
Ruth is-female. (3)

Mike married-to Karen.
Bob married-to Beth.
Jim married-to Liz.

Bob father-of Peg.
Bob father-of Tom.
Bob father-of Jean.
Jim father-of Ruth.
Fred father-of Bill.

Using these facts, the remaining relationships in **R** may be defined as rules. For example,

$$\begin{array}{rll} X \textbf{ mother-of } Y & \text{IF} & Z \textbf{ married-to } X \text{ AND} \\ & & Z \textbf{ father-of } Y, \\ X \textbf{ child-of } Y & \text{IF} & Y \textbf{ mother-of } X \text{ OR} \\ & & Y \textbf{ father-of } X \\ X \textbf{ son-of } Y & \text{IF} & X \textbf{ child-of } Y \text{ AND} \\ & & X \textbf{ is-male}, \\ X \textbf{ daughter-of } Y & \text{IF} & X \textbf{ child-of } Y \text{ AND} \\ & & X \textbf{ is-female}, \end{array} \tag{4}$$

where X, Y, and Z are variables. The bodies of these rules (i.e., the part to the right of IF) consist of two conditions, each of which could be a fact or another rule. These conditions are then connected by the logical operators AND or OR. In general, a rule could have any number of conditions, and a condition could have a logical NOT operation performed on it. For example, the **daughter-of** rule could be modified to use the **son-of** rule by defining it with

$$\begin{array}{rll} X \textbf{ daughter-of } Y & \text{IF} & X \textbf{ child-of } Y \text{ AND} \\ & & \text{NOT } X \textbf{ son-of } Y. \end{array} \tag{5}$$

To satisfy a rule, there must be at least one data value for all variables for which all conditions are simultaneously true. In the **mother-of** rule, there must be at least one value each for X, Y, and Z so that Z is both married to X and the father of Y. Using the format of Eq. (4), additional relationships such as **sister-of** and **brother-of** are straightforward to define. Together, the facts in Eq. (3) and the rules in Eq. (4) form the knowledge base.

In general, a query into a knowledge base consists of a rule with at least one variable. For example, a possible query of this knowledge base could be "Who is the mother of Jean?", which can be expressed as

? **mother-of** Jean, (6)

where ? represents the desired unknown data object. From the knowledge base, only the assertion Beth **mother-of** Jean is true. Hence the conclusion of Eq. (6) is that the query is true when ? is the data object Beth.

Given a query and knowledge base, conclusions can be inferred using either inductive or deductive reasoning. In the inductive case, conclusions of a general nature are inferred by the application of specific queries to the knowledge base. The cardinality of the set of induced conclusions could in general be quite large, and, in principle, conclusions not representative of the knowledge base would be possible.

On the other hand, deductive reasoning produces specific conclusions from a set of general rules and facts, and the conclusions are always a subset of the knowledge base. For simplicity and practicality, we shall limit the allowed conclusions to the data objects within the knowledge base. Therefore, in this paper, we will consider only machines based on deductive reasoning.

Block diagrams for two general architectures of a deductive inference machine are shown in Figs. 2 and 3. Both systems have in common a knowledge base, controller, and inference filter. The functions of the controller are to (1) control the flow of information through the inference machine, (2) accept queries as input from the operator, and (3) transmit conclusions to the operator as output. The knowledge base stores all the data objects and relationships in the form of facts and rules. The role of the inference filter is to generate a set of all conclusions possible given a set of rules and facts from the knowledge base.

The system in Fig. 2 corresponds to a rule-driven inference machine, whereas that in Fig. 3 represents a query-driven inference machine. The systems are distinguished from each other by the methods they employ to infer the conclusions. In the rule-driven system, all possible assertions and facts from the knowledge base are generated *ab initio*, and thereafter the conclusions are derived from these inferences by application of the query. In contrast, the query-driven system first uses the query to select appropriate subsets of the rules and facts and then infers specific conclusions from these rules and facts.

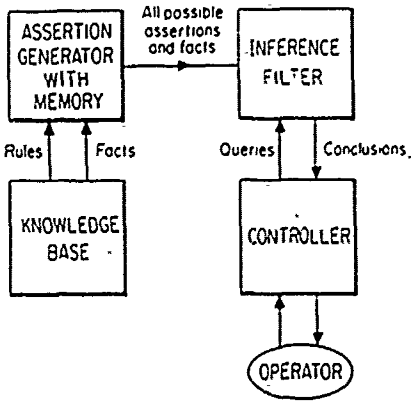


Fig. 2. Block diagram of a deductive rule-driven inference machine.

The rule-driven system of Fig. 2 approaches the ideal parallel system in that the assertion generator produces the facts and all possible assertions from the entire knowledge base by replacing all the rules with appropriate assertions. In the previous example, the **mother-of**, **child-of**, **son-of**, and **daughter-of** rules would lead to the assertions

Beth **mother-of** Peg,
Beth **mother-of** Tom,
Beth **mother-of** Jean,
Liz **mother-of** Ruth,

Tom **son-of** Bob.
Tom **son-of** Beth.
Bill **son-of** Fred. (7)

Peg **child-of** Bob,
Peg **child-of** Beth,
Tom **child-of** Bob,
Tom **child-of** Beth,
Jean **child-of** Bob,
Jean **child-of** Beth,
Ruth **child-of** Jim,
Ruth **child-of** Liz,
Bill **child-of** Fred,

Peg **daughter-of** Bob.
Peg **daughter-of** Beth.
Jean **daughter-of** Bob.
Jean **daughter-of** Beth.
Ruth **daughter-of** Jim.
Ruth **daughter-of** Liz.

Thus the output of the assertion generator would be the set of facts and assertions defined by Eqs. (3) and (7). Note that the knowledge base is not updated by the assertion generator and that the output produced by the assertion generator is computed only once.

As shown in Fig. 2, the assertion generator of the rule-driven machine transfers the entire set of facts and assertions to an inference filter whose function is to match the queries from the controller with the facts and assertions to determine the data objects which satisfy the queries. After it has determined the conclusions for the query, the inference filter transfers the conclusions to the controller for output to the operator

In the example ? **mother-of** Jean, the inference filter would compare the facts and assertions defined by Eqs. (3) and (7) with the query given in Eq. (6) Realizing that ? is the desired variable, the filter would find a match between the query with the third assertion given in Eq. (7) to obtain the answer Beth. In this example, there was only one possible conclusion, but, in general, several data objects may satisfy a query.

In contrast, the query-driven system of Fig. 3 is a more sequential machine than the rule-driven system of Fig. 2. Given a query from the operator, the controller uses the rules associated with the query to select subsets of rules and facts from the knowledge base that are relevant to the query. In the example of Eq. (6), the **mother-of** rule is associated with the query. The controller would examine the **mother-of** rule as defined in the knowledge base and extract its condition relationships **married-to** and **father-of**.

Once it has obtained the necessary subsets of rules and facts, the controller transfers these subsets to the inference filter along with the known data objects from the query [Jean in Eq. (6)]. The inference filter then matches the rules with the known query data to infer the set of data objects which make the query true [Beth for Eq. (6)]. Finally, the inference filter sends the

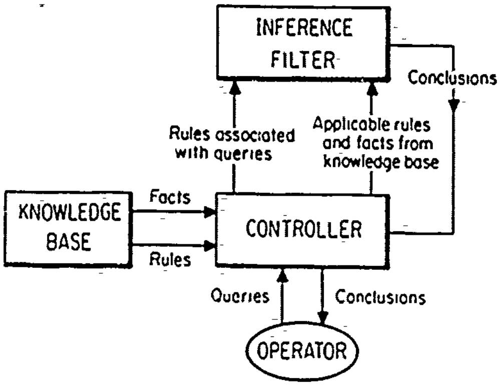


Fig. 3. Block diagram of a deductive query driven inference machine.

conclusions back to the controller for output to the operator.

When consideration is given to implementation of an inference machine, the query-driven system may appear more attractive than the rule-driven system. This is because inferring the possible assertions and storing all the possible assertions and facts in the rule-driven system could be inefficient, expensive, and difficult to realize, particularly for rules which are recursively defined (i.e., when the rule has itself as a condition). Consequently, only query-driven systems are considered in the remaining sections.

III. Hybrid Optical Realizations

We shall confine our discussion to optical inference machines that complement the electronic computer. A complete system, therefore, will be hybrid in nature. This places design constraints on the input and output interfacing devices of the optical system. The optimum designs, therefore, are those that most effectively combine the individual strengths of optics and electronics. Two query-driven designs are described below, the first of which uses matched-filter logic in the inference filter, whereas the second is based on mapped-template logic.

In these systems, the parallelism and speed of optics are exploited to perform the functions of searching, matching, and logic. The role of the electronics is to perform information storage and retrieve and transfer data, rules, and operator queries to the optical processor. Thus in Fig. 3 the inference filter is the optical processor, while the controller and knowledge base constitute the electronic support system.

To implement these optical inference machines, three types of optical devices are required: (1) an input interfacing device which converts electrical signals to 2-D optical signals; (2) an optical logic device; and (3) an output interfacing device for transforming optical signals into electrical signals. The input interfacing device and optical logic device should exhibit at least short-term storage.

In the specific systems discussed below, the electrical-to-optical input device could be any 2-D electrically addressed spatial light modulator (E-SLM) which has short-term storage, such as the e-beam MSLM.[20] An example of an optical logic device which can perform 2-D logic with memory is the photo MSLM,[21,22] which is an optically addressed spatial light modulator (O-SLM). The logic operations that can be performed internally by the photo MSLM include AND, OR, NAND, NOR, XOR, and NOT. The optical-to-electrical output device is a 2-D photodetector array. To obtain good noise rejection and low error rates, digital optical signals (binary intensity levels) are assumed for all input and output signals in the optical processor.

A. Matched-Filter Optical Inference Machine

The general matched-filter optical inference machine employs analog pattern recognition techniques and parallel optical logic to apply a set of given rules to a set of facts to infer a set of logical conclusions to the queries. This method is similar to the optical correlograph system described by Willshaw and Longuet-Higgins.[10]

Figure 4 shows a specific implementation of a query-driven matched-filter hybrid optical inference machine. This machine consists of an electronic controller, two E-SLMs, two O-SLMs, and a photodetector array which is operated in a thresholding mode. In this and subsequent figures, it should be noted that (1) the input light to the O-SLM is absorbed within the device and is not transmitted, and (2) the readout light is reflected out of the device by an internal mirror.

In the matched-filter system of Fig. 4, the facts and rules are grouped in block form (subsets) and stored electronically in the controller for rapid retrieval and transfer to the optical system. The two E-SLMs, O-SLM 1, the lenses L_1, L_2, and L_3, and the photodetector array are arranged to form a classical VanderLugt matched-filter system.[23] Thus lens L_1 is one focal length away from the planes P_1 and P_2, lens L_2 is one focal length away from planes P_3 and P_4, and lens L_3 is one focal length away from planes P_3, P_5, and P_6. The multiplication of the Fourier transforms of the signals

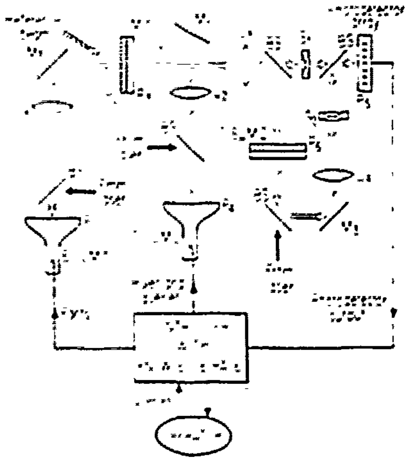


Fig. 4. Matched filter optical inference machine.

to be matched is performed in O-SLM 1, and the matched-filter output is recorded on the photodetector array (shutter S_1 open, S_2 closed). The photodetector then transfers its output to the controller.

If the query dictates that several rules must be applied to the facts in succession, the resulting matched-filter outputs can be combined by using the optical logic capabilities of O-SLM 2. With S_1 closed and S_2 open, the logic output of O-SLM 2 can be imaged onto the photodetector array using lens L_4 and the photodetector output fed back to the controller. This ability permits rules to be applied as many times as necessary to various subsets of facts to generate the logical conclusions.

When operating the matched-filter optical inference machine, the operator queries the system through the electronic controller. In response, the controller writes the applicable subsets of facts onto E-SLM 1 and the applicable subset of rules onto E-SLM 2. This information is coded as a set of predetermined 2-D binary-level patterns. In the query example of Eq. (6), the **mother-of** rule and the complete set of facts in Eq. (3) would be the applicable sets.

The controller then activates O-SLM 1 which holographically records the Fourier transform of the facts as formed by lens L_1. The rules are similarly transformed by lens L_2, and this transform is used to read out O-SLM 1 via mirror M_1 as shown in Fig. 4. The output of O-SLM 1 is transformed by lens L_3 to form the matched-filter output on the photodetector array. This output consists of a set of focused spots of light which indicates the positions of the matches. These signals are then stored in O-SLM 2 and/or fed back to the controller, which then uses this input to select the possible conclusions from the set of facts.

Several options exist at this point, depending on the nature of the query being solved. For example, the controller could now load another part of the query into E-SLM 2, perform a second matched-filtering operation, and with S, closed and S_2 open perform a logical AND (with O-SLM 2) of the second correlation and the first which is already stored in O-SLM 2. The output of O-SLM 2 would then be read out onto the photodetector array. Thus the matched-filter inference machine is capable of sequentially performing all combinations of 2-D optical pattern correlations and binary level logic operations on patterns representing the data objects, rules, facts, and queries.

To solve the ? **mother-of** Jean query in Eq. (6), this system would first examine the query for the specified data objects (Jean in this case) and would then treat the **mother-of** rule as if its variables were replaced by the appropriate data objects. In this case, the effective **mother-of** rule would become

$$\begin{aligned} ?\ \text{mother-of Jean IF}\quad & Z\ \text{married to ? AND} \\ & Z\ \text{father-of Jean.} \end{aligned} \qquad (8)$$

Comparing Eq. (8) with the original **mother-of** rule as defined in Eq. (4), the variables X and Y have been replaced with the desired unknown symbol ? and the data object Jean, respectively. Since the **mother-of** rule has two conditions, the controller has to invoke two matched-filtering operations.

The order in which the conditions are satisfied does not matter since all of them must be true for the query to be satisfied. Since the second condition uses the data object Jean as a constraint, the first matched-filtering operation matches the **father-of** facts (placed on E-SLM 1) with the data object Jean (placed on E-SLM 2). The output of the matched-filter is then a representation of all facts associated with the condition **father-of** Jean. In this case, there is only one fact associated with this condition, Bob **father-of** Jean. The controller then retrieves the father's name Bob and matches the condition Bob **married-to** with the set of facts. The second matched-filter output points to the fact Bob **married-to** Beth. Finally, the controller simply associates the conclusion Beth with ? and returns the conclusion to the operator.

In the case where there are several matches, it is possible for the controller to match all the resulting conclusions with the next condition for full parallelism. Furthermore, if no match is made (i.e., no spots of light above threshold on the photodetector array), the condition cannot be satisfied, making the query false.

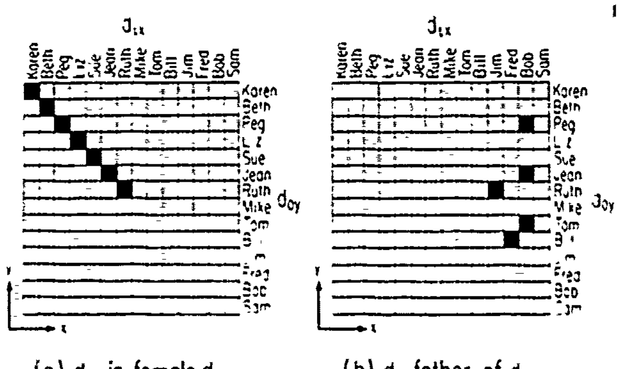
The block electronic storage scheme suggested here is not the most efficient means of storing the rules and the facts because a single data object may be associated with several different facts. However, because electronic storage is relatively inexpensive, block-form storage does not appear to be inappropriate for the initial investigations of these machines.

Since data objects are not expected to change often, partitioning the knowledge base into blocks will generally not have to be done frequently. The advantage of block electronic storage is that it not only reduces the data acquisition and retrieval time but also eliminates the need to transfer the entire knowledge base to the spatial light modulators which currently have only modest space–bandwidth products.

B. Mapped-Template Optical Inference Machine

In the mapped-template optical inference machine mapping templates are used to store the relationships between the data objects and are thus defined by the facts. Conclusions are inferred to queries by applying these mapping templates to the data objects in the order prescribed by the rules. This usage of mapping templates is similar to the associative nets described by Willshaw and Longuet-Higgins.[10]

Using the example defined by Eqs. (1)–(6), the relations **is-male, is-female, married-to,** and **father-of** from the facts in Eq. (3) would map an input set from **D**, the set of data objects defined in Eq. (1), to an output set, also from **D**. Let $\mathbf{D}_i$ and $\mathbf{D}_o$ represent the input and output sets of data objects. Furthermore, let the data objects in the mth position of $\mathbf{D}_i$ and $\mathbf{D}_o$ be denoted by d_{im} and d_{om}. Using the data set D for D and $\mathbf{D}_o$ as defined in Eq. (1), the mapping templates corresponding to the **is-female** and **father-of** facts as defined in Eq. (3) are shown in Fig. 5 with the elements



(a) d_{ix} is-female d_{oy} (b) d_{ix} father-of d_{oy}

Fig. 5. Mapping templates for (a) the is-female facts and (b) the father-of facts for the entire set of data objects in Eq. (1).

of the input set $\mathbf{D}_i(d_{ix})$ along the columns (x axis) and the elements of the output set $\mathbf{D}_o(d_{oy})$ along the rows (y axis) of the templates.

The mapping templates are binary masks consisting of transparent squares (logical 1 and shown as black squares in Fig. 5) on an opaque background (logical 0 and shown as white in Fig. 5). The interpretation of these templates is as follows: A transparent square in the (x, y) position of, say, **father-of** indicates the fact

$$d_{ix} \textbf{ father-of } d_{oy}. \tag{9}$$

Given these two templates, the mapping templates for **is-male** and **married-to** are straightforward to generate.

Note that the mapping between D_i and D_o is not necessarily one-to-one. However, a mapping template is reciprocal in that if the right-hand side of Eq. (9) is specified instead of the left, the relationships for the left-hand side may be inferred from the template.

Alternatively, to limit the size of the mapping template and conserve space, **D** could be subdivided into subsets whose data objects are related in some way. Considering the facts in Eq. (3), it is reasonable to split D into a set of males and a set of females denoted by

$$\begin{aligned} \mathrm{D}_m &= \{\text{Mike, Tom, Bill, Jim, Fred, Bob, Sam}\} \\ \mathrm{D}_f &= \{\text{Karen, Beth, Peg, Liz, Sue, Jean, Ruth}\}, \end{aligned} \tag{10}$$

where $\mathbf{D}_m$ and $\mathbf{D}_f$ represent the male and female sets, respectively. With these subsets, the relationships **is-male** and **is-female** would no longer be needed.

With the data set partitioned, the mapping templates for the factual relationships between the elements of $\mathbf{D}_m$ and $\mathbf{D}_f$ would simply be the corresponding regions in the original full-size mapping templates in Fig. 5. For the relation **is-female** and the data set $\mathbf{D}_m$, the template would always be opaque.

To perform logical inferring, the mapping-template concept is implemented as illustrated in Fig. 6. Given an input vector $\mathbf{D}_i$, the associated output vector $\mathbf{D}_o$ for a particular mapping template is found by first vertically expanding $\mathbf{D}_i$ along the y axis so it forms an array, each row of which equals $\mathbf{D}_i$, as shown in Fig. 6. This expanded form of $\mathbf{D}_i$ is then optically overlaid with the mapping template using imaging optics and a 2-D logical AND operation is performed. The resulting output, when viewed along the rows, corresponds to the output vector $\mathbf{D}_o$.

To perform the reciprocal operation of the mapping template, the input vector would be expanded horizontally and logically ANDed with the mapping template. The output vector would then be taken looking down the columns.

Depending on the mapping template, it is possible for multiple inputs in $\mathbf{D}_i$ to produce the same output element in $\mathbf{D}_o$. For this reason, a 2-D output photodetector array is used for establishing the exact input-to-output correspondence, should this be needed in solving the query.

A hybrid optical inference machine which implements mapped-template logic is shown in Fig. 7. It consists of an electronic controller, two E-SLMs, two O-SLMs, and a 2-D photodetector array. Like the matched-filter optical inference machine, the controller in this system electronically stores the knowledge base and controls the SLMs and the shutter. The modulator O-SLM 1 is operated in the logic mode and usually performs the AND operation, while O-SLM 2 is used as a 2-D memory unit to allow further processing of the outputs, and is optional.

When the controller is given a query by the operator, a vertical line is written on E-SLM 1 at the location of

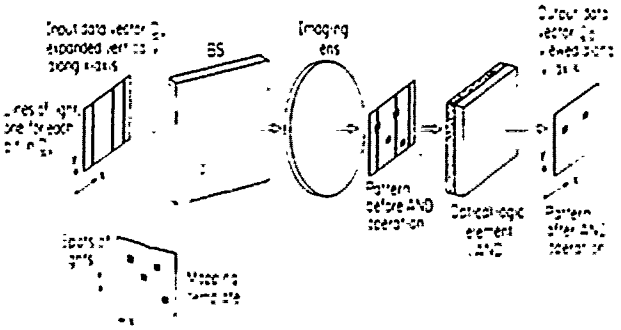


Fig. 6 Conceptual implementation of mapped-template logic

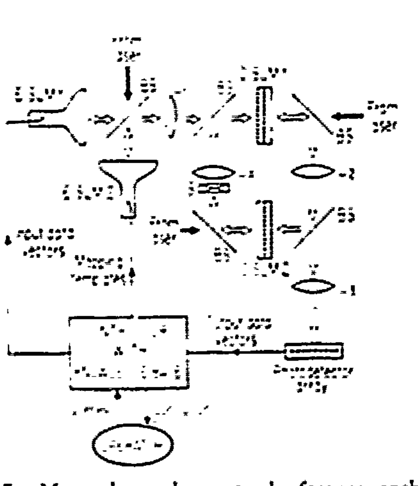


Fig. 7. Mapped-template optical inference machine

the known data objects in $\mathbf{D}_i$. Then the controller writes the mapping template corresponding to the rule (or first condition) associated with the query onto E-SLM 2. The outputs of both E-SLMs are imaged onto O-SLM 1 with lens L_1. The logical AND of the two inputs is formed in O-SLM 1 and imaged onto the photodetector array by lenses L_2 and L_3. If desired, the output could also be imaged onto O-SLM 2 by lens L_2 and latched. The stored output in O-SLM 2 could then be imaged via lens L_4 back into O-SLM 1 by opening shutter S should further processing be necessary.

The output of the photodetector array is fed back to the controller where the inferred data objects in $\mathbf{D}_o$ which satisfy the current mapping rule are determined. Further mapping templates are then applied by the controller as determined by the query and rules.

Operation of this optical inference machine can be demonstrated for the ? **mother-of** Jean query in Eq. (6). As with the matched-filter machine, the mapped-template system considers the effective form of the **mother-of** rule given the data object Jean as specified in Eq. (8). The controller first uses the mapping rule template for **father-of** as shown in Fig. 5 and the input vector corresponding to Jean, which is, from Eq. (1), [0 0 0 0 0 1 0 0 0 0 0 0 0 0]. Since Jean is specified on the output side of **father-of**, the input vector is expanded horizontally rather than vertically on E-SLM 1. Scanning the rows of the output array produces the output vector [0 0 0 0 0 0 0 0 0 0 0 0 1 0] which corresponds to Bob.

The controller then feeds this output vector back to E-SLM 1 as the input vector for the **married-to** mapping template. Since this input is on the right side of the **married-to** rule, the vector is expanded vertically on E-SLM 1. The inference operation is repeated with the **married-to** mapping template on E-SLM 2, producing the output vector (view along the columns) [0 1 0 0 0 0 0 0 0 0 0 0 0 0], which indicates the conclusion Beth.

Since multiple outputs for the same data object could be generated, viewing the rows or columns of the output array could lead to an integral multiple of a single light beam intensity. In this case, the photodetector output is electronically clipped to the single light beam level if the photodetector output is to be fed back to E-SLM 1 as input via the controller.

If the optional optical feedback loop is not used, there is a possible modification to the system in Fig. 6 which will simplify the device requirements. Instead of performing the logical AND operation in O-SLM 1, the output of E-SLM 1 (the expanded input data vector) could be used to read out the mapping template in E-SLM 2, thus eliminating the need for a 2-D optical logic device. However, the advantage of having O-SLM 1 is that (1) it can conveniently perform the logical NOT operation on a condition, and (2) the processed patterns are automatically latched into O-SLM 1. This allows the controller to begin setting up the next mapping template while it simultaneously reads the photodetector array, thus providing some degree of concurrent operation.

Further possibilities for increasing processing speed are to place multiple mapping templates which are spatially separated from each other on E-SLM 2. The input data vectors on E-SLM 1 would have to be repositioned accordingly. However, multiple inferences could then be made in parallel.

IV. Concluding Remarks

Basic architectures for a hybrid optical machine capable of solving symbolic logic problems have been discussed in general terms. This inference machine was considered from both a rule-driven and query-driven approach. Two hybrid optical designs of a query-driven inference machine were described which used matched-filter logic and mapped-template logic.

In comparing the two designs, the mapped-template system should be less demanding on the spatial resolution characteristics of the spatial light modulators and should be easier to implement than the matched-filter machine. Furthermore, the mapped-template system should have better noise performance since there is no analog processing in this system. That is, all optical signals remain encoded as binary intensity levels in the mapped-template system, whereas the matched-filter system must contend with the noise from the analog matched-filtering process, even though binary intensity input and output patterns are used.

Although two hybrid architectures have been presented, other equally effective system designs are possible. Given the growing interest in integrating symbolic logic processing into the computer of the future, the idea of downloading the inference operations of scanning, searching, and matching to a parallel optical processor merits continued investigation.

This work was supported in part by the Air Force Office of Scientific Research under grant AFOSR-84-0358.

References

1. D. S. Nau. "Expert Computer Systems," IEEE Comput. 63 (Feb 1983).
2. T. Moto-oka and H. S. Stone. "Fifth-Generation Computer Systems. A Japanese Project," IEEE Comput. 6 (Mar. 1984)
3. K. L. Clark and F. G. McCabe. *Micro-Prolog: Programming in Logic* (Prentice-Hall, Englewood Cliffs, NJ, 1984).
4. W. F. Clocksin and C. S. Mellish. *Programming in Prolog* (Springer-Verlag, 1981) New York.
5. D. Gabor. "Associative Holographic Memories," IBM J. Res. Dev. 156 (1969).
6. H. Akahori and K. Sakurai. "Information Search Using Holography," Appl. Opt. 11, 413 (1972).
7. M. Nakajima, T. Morikawa, and K. Sakurai. "Automatic Character Reading Using a Holographic Data Processing Technique," Appl. Opt. 11, 362 (1972).
8. A. W. Lohmann and H. W. Werlich. "Holographic Production of Spatial Filters for Code Translation and Image Restoration," Phys. Lett. A 25, 570 (1967).
9. D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. "Non-Holographic Associative Memory," Nature London 222, 960 (1969).

10. D. J. Willshaw and H. C. Longuet-Higgins, "Associative Memory Models," Machine Intell. 5, 351 (1970).

11. R. A. Athale, W. C. Collins, and P. D. Stilwell, "High Accuracy Matrix Multiplication With Outer Product Optical Processor," Appl. Opt. 22, 368 (1983).

12. R. P. Bocker, "Optical Digital RUBIC (Rapid Unbiased Bipolar Incoherent Calculator) Cube Processor," Opt. Eng. 23, 26 (1984).

13. Y. Z. Liang and H. K. Liu, "Optical Matrix–Matrix Multiplication Method Demonstrated by the Use of a Multifocus Hololens," Opt. Lett. 9, 322 (1984).

14. H. J. Caulfield, W. T. Rhodes, M. J. Foster, and S. Horvitz, "Optical Implementation of Systolic Ar-ay Processing," Opt Commun. 40, 86 (1981).

15. M. Carlotto and D. Casasent, "Microprocessor-Based Fiber-Optic Iterative Optical Processor," Appl. Opt. 21, 147 (1982)

16. A. Huang, "Parallel Algorithms for Optical Digital Computers," Proc. Soc. Photo-Opt. Instrum. Eng. 422, 13 (1983).

17. K. Brenner and A. Huang, "An Optical Processor Based on Symbolic Substitution," in *Technical Topical Meeting on Optical Digest, Computing* (Optical Society of America, Washington, D.C., 1985), paper WA4.

18. A. Huang, "Why Use the Parallelism of Optics?," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, D.C., 1985), paper WA2.

19. A. D. Fisher, C. L. Giles, and J. N. Lee, "An Adaptive, Associative Optical Computing Element," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, D.C., 1985), paper WB4.

20. A. Schwartz, X. Y. Wang, and C. Warde, "Electron-Beam-Addressed Microchannel Spatial Light Modulator," Opt. Eng. 24, 119 (1985).

21. C. Warde, A. M. Weiss, A. D. Fisher, and J. I. Thackara, "Optical Information Processing Characteristics of the Microchannel Spatial Light Modulator," Appl. Opt. 20, 2066 (1981)

22. C. Warde and J. I. Thackara, "Oblique-Cut $LiNbO_3$ Microchannel Spatial Light Modulator," Opt. Lett. 7, 344 (1982)

23. A. B. VanderLugt, "Signal Detection by Complex Spatial Filtring," IEEE Trans. Inf. Theory IT-10, 2 (1964).

9 Trends in Knowledge Base Processing Using Optical Techniques [11]

Reprinted from PROCEEDINGS OF THE 1989 IEEE INTERNATIONAL CONFERENCE
ON SYSTEMS, MAN, AND CYBERNETICS, Cambridge, MA, November 14-17, 1989

TRENDS IN KNOWLEDGE BASE PROCESSING USING OPTICAL TECHNIQUES

James A. Kottas and Cardinal Warde

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Cambridge, Massachusetts, 02139

ABSTRACT

Knowledge base systems (KBS's) are becoming increasingly more important for many scientific and engineering applications. Over the past few years, several researchers have considered optics for implementing KBS's in an attempt to capitalize on the potential speed and parallelism. This paper presents a review of recent research efforts along with a discussion of the relative merits and limitations of using optics as an implementation technology. To a large extent, past efforts have focused on (1) representing knowledge using matrix-like formalisms and (2) designing system architectures based on optical inner product processors (such as matrix-vector multipliers) and optical correlators. Actual implementations are impeded primarily by the limitations of current spatial light modulators. New directions include the use of symbolic substitution and neural network ideas.

INTRODUCTION

A knowledge base system (KBS) manipulates symbolic information to produce useful output conclusions given input queries or requests. Three familiar examples are database managers, relational database systems, and inference machines (such as expert systems). The knowledge base consists of sets of symbols and relationships between them. The allowed operations are determined by the type of KBS being considered. For example, if the knowledge base is a database (or relational database) containing records of information, typical operations include sorting and searching on specific fields within a database record. For an inference machine, the knowledge base is a collection of symbols and relationships arranged as sets of facts (axioms) and rules Besides searching, a fundamental operation of such a KBS is inference (usually deductive). If the facts and rules are focused on a specific area of knowledge, the KBS is known as an expert system.

Conventionally, KBS's have been implemented in software on electronic computers. The primary languages have been LISP, PROLOG, and with specialized hardware, PARALOG [1,2]. With the software approach, the knowledge base and its associated operations are programmed into the KBS. More recently, a connectionistic (neural network) approach has been applied to an inference machine KBS to realize a trainable expert system [3]. This method allows facts and rules to be learned by a KBS using a train by example procedure on known sets of input queries and output conclusions.

While software based KBS's have been quite adequate for small to medium size knowledge bases, their performance can be degraded significantly by large and especially very large knowledge bases. To process vast amounts of information, a large, fast memory that can be searched efficiently is required. Herein lies the potential of an optical implementation of a KBS.

Optics offers a high degree of parallelism with its natural two dimensional data path. This allows for efficient parallel searching techniques, particularly when holographic storage methods are employed. Furthermore, since light beams can intersect with negligible interaction, large numbers of interconnections between processing components can be made with more flexibility than with electronic wires. To illustrate the severity of the wiring problem, consider an optical data path consisting of 1000 × 1000 pixels. Forming interconnections between two planes with this format is relatively straightforward for an optical system. However, electronically interconnecting the 10^6 locations in one plane with the 10^6 locations in the second plane could require 10^{12} wires in the worst case (i.e., fully interconnected). Such a high wiring density is effectively impractical for current VLSI (Very Large Scale Integration) technology.

In this paper, we present an overview of the basic approaches being considered for realizing optical KBS's. The discussion begins with a summary of the techniques for representing symbolic information optically and continues with a survey of the available optical hardware. Then, descriptions of selected optical architectures are given, followed by a discussion of the relative strengths and limitations of these optical systems. The interested reader is referred to Ref. 4 for another view on the impact of optics on KBS's.

OPTICAL KNOWLEDGE BASE SYSTEMS

Representation of Symbolic Information

One of the fundamental design criteria for a KBS is how

CH2809-2/89/0000-1250 $1.00 © 1989 IEEE

<table>
<tr><td colspan="2">Corporation Name</td><td rowspan="4">Logo</td></tr>
<tr><td colspan="2">Address</td></tr>
<tr><td>City</td><td>State</td></tr>
<tr><td>Zip Code</td><td>Country</td></tr>
<tr><td>Telephone</td><td colspan="2">Slogan</td></tr>
</table>

Figure 1 Example field structure for a corporation database record.

Corp. 1	Corp. 2	Corp. 3
Corp. 4	Corp. 5	Corp. 6
Corp. 7	Corp. 8	Corp. 9

Figure 2. Example record structure for a corporation database. Each corporation has a record entry like that shown in Fig. 1.

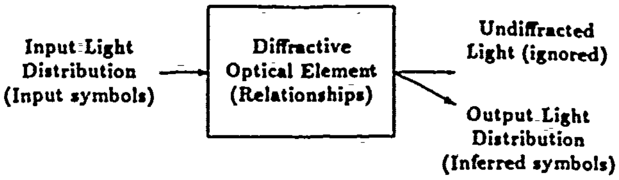

Input Light Distribution (Input symbols)
Diffractive Optical Element (Relationships)
Undiffracted Light (ignored)
Output Light Distribution (Inferred symbols)

Figure 3. Diffractive-type memory for storing the knowledge base.

Input Symbols

$$\begin{array}{ll|cccc|}
 & & A & B & C & D \\
\hline
 & A & 0 & 0 & 1 & 0 \\
\text{Output} & B & 1 & 0 & 0 & 0 \\
\text{Symbols} & C & 0 & 0 & 0 & 0 \\
 & D & 0 & 1 & 1 & 1 \\
\hline
\end{array}$$

Relationships
$A \rightarrow B$
$B \rightarrow D$
$C \rightarrow A, D$
$D \rightarrow D$

Figure 4. Encoding relationships between symbols using a binary relationship matrix.

symbolic information is represented. In a database application, the database usually is organized as an array of records, each of which is decomposed into a set of fields. For example, in a database of corporations, each record could have fields such as corporation name, address, telephone number, etc. In an optical database system, the fields could be arranged in a suitable spatial pattern such as that shown in Fig. 1, and this pattern would constitute a record within the database. An array of records could be organized into a matrix as illustrated in Fig. 2, thus resulting in a two dimensional storage format for the database.

In an optical inference machine, both the basis symbols of information and the relationships between them (which form the facts and rules) are encoded in the knowledge base. There are primarily two methods for storing this information. diffractive type memories and relationship matrices. As shown in Fig. 3, a diffractive type memory is an optical subsystem that processes an input light distribution using diffraction to produce an output light distribution. The input distribution represents either (1) the symbol, (2) a collection of symbols, or (3) a portion of a symbol. The diffracted output then corresponds to the associated or inferred symbol implied by the input symbol in the first two cases or the completed symbol in the third case. Two examples of this type of storage medium are holographic associative memories (both hetero and auto associative) and matched spatial filters (MSF's).

In a relationship matrix, a symbol of information is represented by a row and a column within the matrix. An example format is shown in Fig. 4. The value of a matrix element, T_{ij}, corresponds to the relative amount that symbol s_j implies symbol s_i. In principle, T_{ij} could be a continuous value, but in Fig. 4, it is shown as a binary value. With a relationship matrix, the inference process is very similar to an inner product operation like matrix-vector multiplication. For example, a binary input vector s has $s_j = 1$ for each symbol j that is active (0 otherwise). For a particular relationship matrix T, the set of output symbols implied by the selected input symbols is given by thresholding the matrix-vector product Ts to form a binary output vector. The thresholding operation is necessary to make the input and output vectors compatible so other relationship matrices may be applied to generate further conclusions. The relationship matrix formalism includes the mapping templates described in Ref. 5 and the directed graphs and adjacency matrices in Refs. 6-8.

By combining a diffractive-type memory with a relationship matrix, more flexibility is gained for storing and accessing a knowledge base. This is the approach used by several optical KBS's.

Optical Hardware

There are a variety of general-purpose optical devices that can be used in an optical KBS. Typically, an optical KBS is a hybrid optical/electronic system with the electronics serving to control the optics and to provide an interface between the user and the optics. Consequently, four classes of devices are needed. electrical-to-optical, optical-to-optical, optical-to-electrical, and specialized. Because of its unique coherence properties, the laser is usually the light source of choice and should be assumed below unless otherwise specified.

Electrical-to-optical devices, called electrically-addressed spatial light modulators (E-SLM's), convert electronic [illegible] nals from the controller into suitable optical signals. For one-dimensional (1-D) inputs (vectors), acousto-optic light modulators and linear arrays of light-emitting diodes (LED's) or laser diodes can be used. Two-dimensional (2-D) inputs (matrices) can be realized with commercially available E-SLM's such as the LightMod [9]. With some modification, portable liquid-crystal-display (LCD) televisions also can be employed [10–12].

Optical-to-optical devices, optically-addressed spatial light modulators (O-SLM's), can be used as optical memories to store intermediate conclusions or as active processing elements. Commercially available O-SLM's include the microchannel spatial light modulator (MSLM), the liquid-crystal light value (LCLV), and the Pockels readout optical modulator (PROM) [9]. Depending upon their internal design, these O-SLM's can differ significantly in the types of active processing operations offered. For example, the MSLM can perform logic functions and thresholding and has long-term storage for both analog and binary-level images [13]. By comparison, the LCLV also can implement logic functions and thresholding but only has very short-term storage [14,15].

In addition to these commercially available SLM's, many other SLM's (both electrical-to-optical and optical-to-optical) are currently under development [15], including bistable optical devices (BOD's) [16].

Optical-to-electrical devices (optical detectors) transform a 2-D light distribution into a set of serial or parallel electronic signals that represent the output conclusions of the optical KBS. Because of their usefulness in many other applications, optical-to-electrical devices are the most well-developed of those discussed so far. Examples of these devices include silicon photodetector arrays and charge-coupled-device (CCD) cameras.

Specialized devices perform dedicated tasks that cannot be achieved easily with any SLM. Besides conventional optical components such as lenses, mirrors, and beamsplitters, fixed filter masks can be made from photographic film. Static diffractive elements can be formed using holographic film and dynamic diffractive elements can be realized using photorefractive crystals [17].

Fundamental Optical Architectures

Diffractive-Type Memory The basic architecture for implementing a diffractive-type memory is the correlator, shown in Fig. 5. It consists of two Fourier-transforming lenses, L_1 and L_2, and a diffractive element arranged as a coherent optical processor. The first lens forms the 2-D spatial Fourier transform $\tilde{U}_{in}(u,v)$ of the input light distribution $U_{in}(x,y)$ at the filter plane. Here, a diffractive optical element such as a holographic filter or photorefractive crystal or any other type of optical filter is placed. The transmittance of this element, $H_{fil}(u,v)$, multiplies the transform $\tilde{U}_{in}(u,v)$.

The light distribution exiting the diffractive element, $\tilde{U}_{in}(u,v)H_{fil}(u,v)$, is transformed back from the spatial Fourier domain into the space domain by the second lens to produce the output distribution $U_{out}(x,y)$. If the diffractive element is a matched spatial filter (MSF), the output plane will contain spots of light at the locations in the input plane of the pattern being matched.

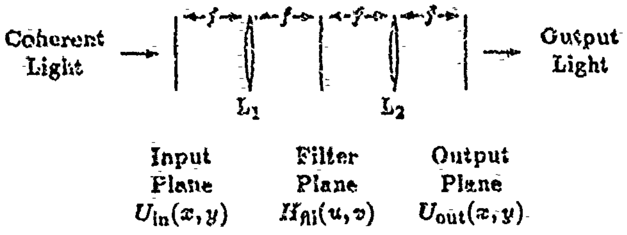

Coherent Light
f
f
f
f
Output Light
L_1
L_2
Input Plane $U_{in}(x,y)$
Filter Plane $H_{fil}(u,v)$
Output Plane $U_{out}(x,y)$

Figure 5: Basic optical correlator (space-invariant).

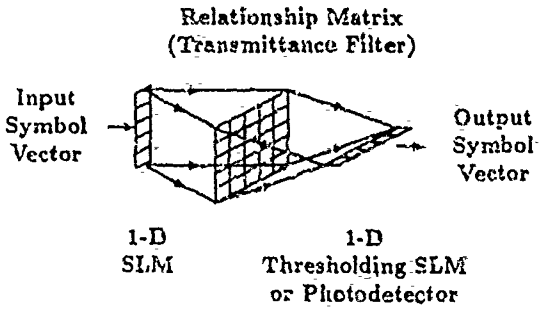


Figure 6. Optical inner product processor for implementing relationship matrices. Expansion optics not shown.

For example, in the database record shown in Fig. 2, if the diffractive element was an MSF of a city name and the database of corporation records was placed in the input plane, then all corporations in that particular city would be indicated by spots of light in the output plane. These light beams would be located at the positions of the "city" field in the matching records. This example illustrates the parallel searching capabilities of optics.

The configuration shown in Fig. 5 has all planes and lenses separated by the focal length f. This setup performs space-invariant processing whereby a shift in the input $U_{in}(x,y)$ only causes the output $U_{out}(x,y)$ to shift accordingly. However, other elements (lenses, filters, etc.) may be added in addition to the distances being varied (appropriately) to make a space-variant processor. In this case, the output is not shift-invariant but will change as the input light distribution is shifted.

Relationship Matrices Since relationship matrices are processed via a matrix-vector inner product with perhaps a nonlinear thresholding on the result, this encoding scheme can be implemented using the basic matrix-vector processor shown in Fig. 6. A vector of light beams representing a set of symbols is presented to the system via a 1-D SLM. Each light beam is spread horizontally using cylindrical or fiber optics (not shown) across a filter mask whose transmittances are related to the values in a relationship matrix. This mask can be implemented using another SLM.

The light then is focused vertically (using cylindrical or fiber optics again) onto a 1-D O-SLM or a linear photodetector array. Any thresholding operation that is needed is performed by this device. Because of the crossed cylindrical (or fiber) optics and the multiplicative transmittance mask, this architecture implements the matrix-vector inner product with complete parallelism.

Depending upon the type of relationship matrix employed, it is also possible to implement an effective inner-product-type operation using the basic optical correlator in Fig. 5. This method usually requires the correlation filter to be encoded in a special way. Therefore, the details of th'3 type of approach only are referenced in the next section.

Optical KBS Research

Several researchers have investigated various aspects of optical knowledge base systems, particularly for inference applications [5-8,18-29]. In this section, several of the approaches taken are summarized in terms of the data representations employed, the focus of the work with respect to optical KBS's, and the relevant optical architectures and issues. The nomenclature used in the literature is retained here and related to the general framework developed above.

The review focuses on inference machines since most of the research efforts have been directed towards this area. In this type of KBS, the system is presented with a query and then conclusions are derived from the relationships in the knowledge base using deductive inference. The conclusions can be either yes/no responses or the set of symbols that satisfy the query.

Warde and Kottas [5] present two architectures for an optical inference machine based on a simplified implementation of PROLOG. Here, the relationships are used to construct a set of facts and rules about data objects (the symbols in Ref. 5). One architecture, shown in Fig. 7, stores the relationships of a knowledge base in a diffractive-type memory and uses an optical correlator as a matched filter to infer new symbols from previous ones. The relationships are stored holographically on O-SLM 1 via E-SLM 1 and can be updated by the electronic controller as needed. The input plane, filter plane, and output plane of the matched-filter correlator are P_1, P_3, and P_5, respectively. Alternatively, plane P_6 at the input to O-SLM 2 could be used as the correlator output plane. In this case, O-SLM 2 can be used to perform additional processing on the image representing the inferred symbols. Examples of such processing include accumulating sequences of inferred symbols and then combining them using thresholding and/or logic operations (AND's, OR's, etc.).

The second architecture, shown in Fig. 8, implements a relationship-matrix form of the knowledge base using optical mapping templates. These templates are binary transmission masks that, when used in an inner product processor, can infer a new vector of symbols from a given vector of symbols The input symbols are written in an expanded form on E SLM 1 by the electronic controller and the mapping templates are stored on E SLM 2. In effect, the templates cause E SLM 2 to become a modifiable cross bar switch. The inner product is formed by using the expanded vector of input symbols from E-SLM 1 to read out the current mapping template on E-SLM 2. O-SLM 1 thresholds the output which then can be latched into O-SLM 2 for further processing or accumulation of results. The electronic controller has a more active role in this architecture than in the matched-

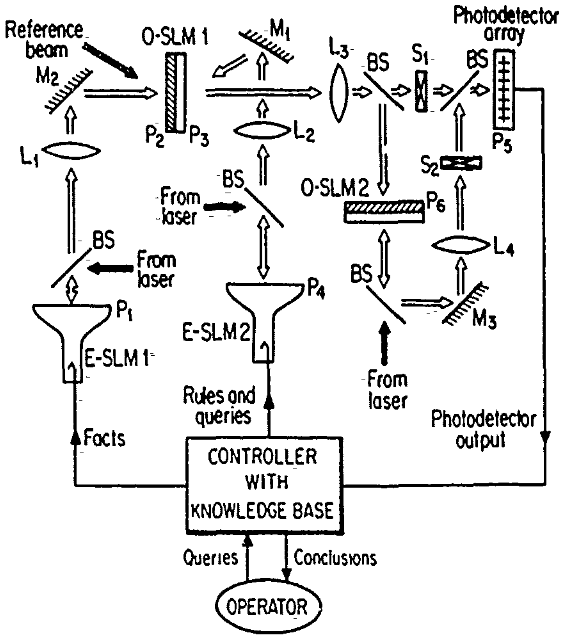


Figure 7: Matched-filter optical inference machine (from Ref. 5).

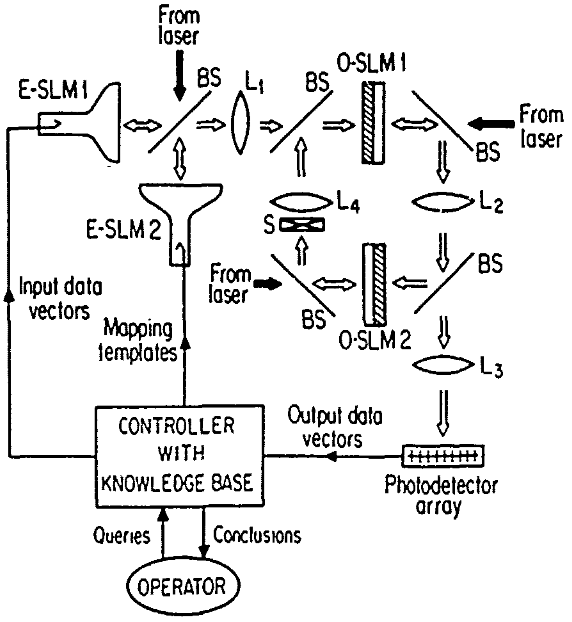


Figure 8: Mapped-template optical inference machine (from Ref. 5).

filter architecture because both the input symbols and mapping templates need to be updated as often as dictated by the inference problem being solved.

Jau *et al.* [8] also have considered optical expert systems from a PROLOG viewpoint and have investigated an approach based on relationship matrices that is similar to the mapping templates of Warde and Kottas [5]. They develop a method for combining binary relationship matrices (fact matrices) using matrix algebra into new relationships, allowing more complex rules to be generated out of the basis set of relationships and symbols (the facts in the knowledge base). However, they extend this formalism by presenting an algorithm for updating the relationship matrices via an update rule when new information is available. The proposed architecture is a general optoelectronic system based on optical matrix-vector and matrix-matrix multipliers.

McAulay [18] uses a probabilistic relationship matrix to develop a forward-inference architecture for a real-time diagnostic expert system. In this type of system, the input symbols are a set of events or conditions and the output symbols are a collection of hypotheses. A query consists of a particular set of input events and the output conclusion is the probability that each hypothesis is true. In the medical expert system described by McAulay, the input conditions are symptoms and the output hypotheses are illnesses.

The architecture is shown in Fig. 9. Binary input symbols (called events in Ref. 18) are presented to the system one at a time on the 1-D SLM (left side of Fig. 9). This input is split between parallel channels, each of which forms an inner product processor. The 2-D SLM's in each channel store a relationship matrix. In one channel, the matrix is the set of *a priori* probabilities that an input event corresponds to a particular outcome, and the matrix in the other channel stores the probability of an event occurring in the absence of the particular outcome.

The outputs of these channels are detected by 1-D CCD's and combined in a set of N parallel processors to form a set of *a posteriori* probabilities. These processors, in conjunction with another inner product processor configured as a doubling summer, compute the updated Bayesian probabilities for each outcome (hypothesis) as each additional event is given as input. Once all events have been processed, the output of the final 1-D CCD array (on the right side of Fig. 9) contains the final probabilities of all outcomes given all the events.

Eichmann and Caulfield [19] consider the same type of problem as McAulay, although in a different context. They present two methods for determining the elements of a relationship matrix which will aid in making decisions. The input symbols are in the form of a binary knowledge vector that contains the answers to several yes/no questions (the events of McAulay [18]). The output is either a binary answer vector (one method) or a set of *a posteriori* probabilities (the second method) indicating the inferred conclusions (which hypotheses are true). Both methods are based on Bayesian principles and optimal Gaussian classifiers, and an algorithm for incrementally updating the relationship matrix elements (the weights) is prescribed. The implied optical architecture utilizes threshold logic units in the conventional optical matrix-vector and matrix-matrix multipliers (e.g., the inner product processor in Fig. 6).

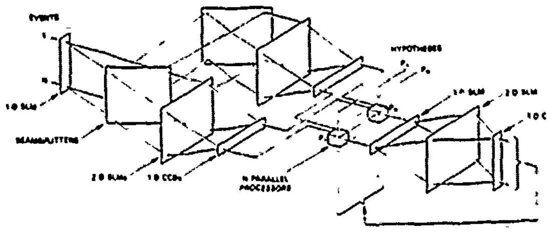


Figure 9: Optical architecture for a real-time diagnostic expert system (from Ref. 18).

Szu and Caulfield [20] employ relationship matrices as associative memories. An interesting aspect of their optical representation for the relationship matrices is that each matrix value is represented by a 2-D binary submatrix rather than a single transmittance. This submatrix allows particular attributes of a symbol to be incorporated into the knowledge base, although it requires more space on an SLM. They propose to input queries using SLM's and to store the relationship matrices in page-oriented holographic memories. The paged memory uses many holograms, each of which stores a subset of the total knowledge base. This method allows the knowledge base to be increased by simply adding holograms to the system.

Haney *et al.* [21] have investigated optical techniques for increasing the efficiency of heuristic searches. They use binary constraint matrices (another form of relationship matrix) as a means of pruning a search tree through the knowledge base before or during the search. In general, a constraint matrix is a binary-valued array indicating a specialized collection of facts which relate multiple sets of symbols. Binary constraint matrices focus on two sets of symbols.

In Ref. 21, Haney *et al.* develop a set of binary constraint matrices using a consistent labeling problem as an example. In this type of problem, there are N units (one set of symbols), each of which is to be assigned one of L labels (the second set of symbols). Let the units be denoted by $u_1, u_2, \ldots, u_N$ and the labels by $l_1, l_2, \ldots, l_L$. Furthermore, let R be a relationship that associates units with labels. Now, a set of facts can be generated in the dyadic form "$u_i\ R\ l_m$" and can be interpreted as "Unit i is associated with label m through relationship R." A constraint matrix $R(i,j)$ [in Ref. 21] is an $L \times L$ matrix with binary elements r_{mn} such that $r_{mn} = 1$ when the facts "$u_i\ R\ l_m$" and "$u_j\ R\ l_r$" are consistent with (i.e., satisfy) other constraints, such as each unit must have a different label. These additional constraints are actually higher level relationships which depend upon the units, the labels, and the basic association relationship R. If $r_{mn} = 0$, both facts above cannot be true simultaneously, and any other conclusions that would depend upon them can be ignored.

With this formalism, the pruning of a search tree is equiv-

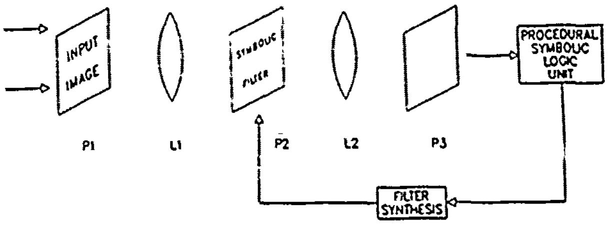


Figure 10: Procedural-based optical inference processor (from Ref. 23).

alent to a forward search through the constraint matrices. This can be done using an optical matrix-vector multiplier. To eliminate the j^{th} unit, the rows of $R(i,j)$ are multiplied by the matrix $R(j,k)$ to form the rows in the new (and stronger) constraint matrix $R'(i,k)$. This process can be repeated to reduce effectively the size of the search tree that needs to be traversed.

Casasent and his colleagues [6,22-25] have investigated the use of knowledge base processing techniques for object recognition, identification, and classification. These researchers utilize an assortment of different architectures based on the optical correlator shown in Fig. 10. The input to this system (at plane P_1) is an image of objects to be recognized rather than an encoded vector or matrix of symbols. A holographic filter containing a set of frequency-multiplexed MSF's for the desired objects is placed in the filter plane (P_2). The spatial carrier frequencies cause the output correlations for the objects to appear on different detectors in the output plane (P_3) [23].

The symbolic logic processor (unit) after plane P_3 is used when various features of the objects are used to make the MSF's instead of the objects themselves. The electronic feedback loop from the symbolic logic processor to the filter plane allows various filters to be synthesized and used to analyze the image. As a result, more structured relationships such as "Object A has all of the features of object B but none of the features of object C" can be processed. Since, in this example, the features for B and C must be examined first before a determination about A can be made, this architecture implements a procedural-like algorithm to object recognition. The symbolic logic processor can be implemented using either an electronic controller or a more sophisticated arrangement of optical correlators.

In the other work [6,22,24,25], Casasent and his colleagues develop other data representations based on directed and relational graphs (both forms of relationship matrices) for performing object recognition using optical knowledge base processing techniques. Furthermore, these authors present alternative architectures such as space- and time-integrating optical processors for directed graphs [6]. In another approach, multiple optical correlators are used to implement a production system (a type of inference machine) using both neural network and symbolic substitution ideas [26]. For further details, the interested reader is directed to the references cited above.

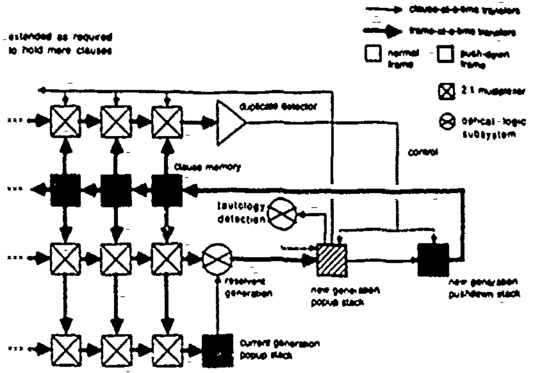


Figure 11. Block diagram of an optical resolution system (from Ref. 29).

A rather novel approach to making inferences from symbolic information is being investigated by Schmidt and Cathey [27-29]. They are examining the use of mathematical resolution to make inferences as a means of avoiding the often-large dynamic data requirements normally found in artificial intelligence problems. This approach is unique in that mathematical resolution is a "proof-by-contradiction" method of inference.

Given a statement whose truth is in question (the query), the process of mathematical resolution proves that the statement is true by showing that the negation of the statement contradicts the axioms (the facts and rules arranged from the relationships and symbols in the knowledge base) This involves accepting an input query in the form of a binary vector and combining it with the literals (the facts or axioms in Refs. 27-29, also represented by binary vectors). The rules of the knowledge base are used to test the resulting vectors for tautologies. These tautology vectors can be reduced further by eliminating nontautological vectors (conclusions that contradict the knowledge base information) and useless or duplicate tautologies. This process may require several iterations.

A block diagram of an optical resolution system is shown in Fig. 11. It combines five different subsystems.

1. A stack memory with parallel access for storing new result vectors.
2. A processor to combine vectors in parallel.
3. A processor to reject nontautological vectors in parallel.
4. A processor to eliminate duplicate tautological vectors in parallel.
5. A controller to monitor the vectors to end the iteration

Schmidt in Ref. 29 presents optical architectures for implementing these subsystems. He has simulated the system on a sample inference problem and compared it to a conventional serial approach and has learned that the identification and elimination of tautologies (step 4 above) is computationally the most significant step of the optical resolution process.

A very different approach to symbolic knowledge processing is considered by Derstine and Guha [7]. They propose

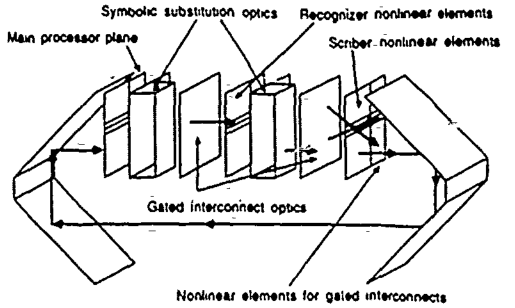

Symbolic substitution optics
Recognizer nonlinear elements
Main processor plane
Scriber nonlinear elements
Gated interconnect optics
Nonlinear elements for gated interconnects

Figure 12: Schematic of the SPARO architecture for an optical finite state machine (from Ref. 7).

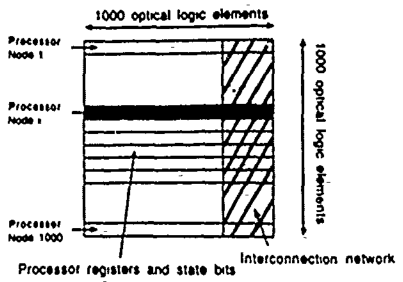

1000 optical logic elements
Processor Node 1
Processor Node i
Processor Node 1000
1000 optical logic elements
Processor registers and state bits
Interconnection network

Figure 13: Optical data path for the SPARO architecture at the main processor plane in Fig. 12 (from Ref. 7).

an optical architecture called SPARO (Symbolic Processing ARchitecture in Optics) to implement a symbolic processing language. Their work, along with that of other researchers already mentioned (Warde and Kottas [5] and Jau *et al.* [8]), was influenced by the PROLOG language. In particular, Derstine and Guha consider PARLOG, a version of PROLOG in which predicates are evaluated in parallel rather than serially.

The SPARO architecture is intended to solve a particular symbolic processing problem, that of combinator graph reduction in pure functional languages such as PARLOG. This process involves reducing a combinator graph (similar to a binary graph) to its simplest form that is consistent with the relationships in the knowledge base.

The SPARO architecture is illustrated in Fig. 12 and the corresponding optical data path in Fig. 13. It is a type of optical finite state machine (OFSM) formed by layers of processing elements (optical logic gates) and optical interconnects. Sets of symbolic substitution optics in conjunction with the interconnections effectively implement the "microcode" between processor elements for performing the graph reduction in parallel. As shown in Fig. 13, the optical data path is decomposed into an array of areas, one for each processor node. A node is a collection of optical bits that make up the state of the node (local memory) and an interconnection register (analogous to a pointer in a software data structure). There is no external or addressable memory system here.

With each iteration around the loop (equivalent to one machine cycle), the processor nodes are updated and the new state information is broadcast to the appropriate destination nodes. In principle, it should be possible to reprogram the system to perform different functions by redesigning the symbolic substitution optics and the interconnects

Alternative Approaches

The idea of using symbolic substitution as a means of doing optical processing is not new. However, the work by Derstine and Guha [7] represents one of the first attempts at using symbolic substitution specifically in an optical KBS. Casasent and Botha [30] also have considered symbolic substitution in the context of multifunctional optical processing systems based on multiple optical correlators. Several other researchers have been working on optical architectures for performing general symbolic substitution [31-34]. Work in the field should be encouraged since this method is another way to implement relationships between symbols.

Furthermore, the theories of connectionistic (neural network) computing offer several opportunities for optical KBS's. There currently is considerable work being done on optical neural networks [35]. Given that neural networks can be trained to perform a wide variety of tasks, this approach has the exciting possibility that the knowledge base can be learned instead of programmed. Botha *et al.* [26] have started to look at a combination of neural networks and symbolic substitution for optical inference systems.

Other approaches include developing alternative data representations. For example, Kottas and Warde [36] are examining various neural network methods for incorporating the time domain into the representations of symbolic information. This is in contrast to the relationship matrix formalism in which the symbols are represented by static positions in the matrices. One potential advantage of this approach is that the system could make use of its state space more efficiently at the expense of slower data access. This could allow larger knowledge bases to be considered using current optical device technology. Preliminary results show that the methods being developed actually could take advantage of any imperfections or irregularities in the optical devices. Nevertheless, different data representation should be considered in future designs of optical KBS's.

Strengths and Limitations

Two of the advantages of an optical KBS implementation result from the use of holographic storage techniques and optical interconnects between processing stages. The holographic storage offers large capacity, fault tolerance, and parallel searching methods. Optical interconnects permit large numbers of connections between pixels in two processing planes with low crosstalk.

On the other hand, the development of practical optical KBS's, particularly inference machines, is limited primarily by the SLM's. Although some are commercially available and many more are under development, current SLM's have relatively low framing rates and resolutions. Thus while the optical propagation delay between devices is negligible, the processing delay within an SLM is significant. This also can cause input/output bottlenecks, particularly in the electrical-to-optical and optical-to-electrical conversions. Because of their limited resolution and finite device size, SLM's can support only small numbers of symbols, either for a database or a knowledge base. As a result, the optical KBS architectures described above are usually slower than their software counterparts. However, when large numbers of symbols can be supported by the SLM's, optical KBS's could become feasible and cost effective.

SUMMARY

Optical knowledge base systems are still in the research stage. In the area of knowledge representation, several encoding schemes have been developed, although most rely on some sort of matrix formalism. In the optics realm, various system architectures have been proposed to implement these encoding schemes. These architectures are based primarily on optical inner product processors (such as matrix-vector multipliers) and optical correlators. Most of them have been investigatated with theoretical analyses and/or computer simulations. To date, few of them have been built using real optical hardware. Because of device limitations, particularly with spatial light modulators, these implementations are restricted to relatively small knowledge bases and simple symbolic processing problems (like forward inference). With improved devices, these architectures could prove to be practical when large knowledge bases are considered.

In the meantime, new methods for representing knowledge and more advanced architectures are needed that can utilize current device technology. Fortunately, alternative paths for research exist through areas such as optical neural networks and optical symbolic substitution techniques. The future offers exciting potential for the development of knowledge base processing systems based on optical implementation technologies.

REFERENCES

1. D. S. Nau, "Expert Computer Systems," IEEE Comput., 63 (Feb. 1983).
2. T. Moto-oka and H. S. Stone, "Fifth-Generation Computer Systems, A Japanese Project," IEEE Comput., 6 (Mar 1984).
3. S. I. Gallant, "Connectionist Expert Systems," Comm. ACM 31, 152 (1988).
4. P. B. Berra, A. Ghafoor, P. A. Mitkas, S. J. Marcinkowski and M. Guizani, "The Impact of Optics on Data and Knowledge Base Systems," IEEE Trans. Know. & Data Eng. 1, 111 (1989).
5. C. Warde and J. A. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Appl. Opt. 25, 910 (1986).
6. D. Casasent and E. Baranoski, "Directed Graph for Adaptive Organization and Learning of a Knowledge Base," Appl. Opt. 27, 534 (1988).
7. M W. Derstine and A. Guha, "Design Considerations for an Optical Symbolic Processing Architecture," Opt. Eng. 28, 434 (1989).
8. J. Y. Jau, F. Kiamilev, Y. Fainman, S. Esener, and S. H. Lee, "Optical Expert System Based on Matrix-Algebraic Formulation," Appl. Opt. 27, 5170 (1988).
9. T. E. Bell, "Optical Computing: A Field in Flux," IEEE Spectrum 23, 34-57 (Aug. 1986).
10. H.-K. Liu, J. A. Davis, and R. A. Lilly, "Optical-Data-Processing Properties of a Liquid-Crystal Television Spatial Light Modulator," Opt. Lett. 10, 635 (1985).
11. M. Young, "Low-Cost LCD Video Display for Optical Processing," Appl. Opt. 25, 1024 (1986).
12. F. T. S Yu, S. Jutamulia, and T. W. Lin, "Real-Time Polychromatic Signal Detection using a Color Liquid Crystal Television," Opt. Eng. 26, 453 (1987).
13. C. Warde, A. M. Weiss, A. D. Fisher, and J. I. Thackara, "Optical Information Processing Characteristics of the Microchannel Spatial Light Modulator," Appl. Opt. 20, 2066 (1981).
14. A A. Sawchuk, "Digital Logic and Computing with Optics," *Optical Computing*, John A. Neff, ed., Proc. SPIE 456, 41 (1984).
15. A. D. Fisher, "A Review of Spatial Light Modulators," *Technical Digest*, Topical Meeting on Optical Computing, Lake Tahoe, Nevada, Optical Society of America, pp. TUC 1-1 to 1-4 (Mar. 1985).
16. H. M. Gibbs, *Optical Bistability: Controlling Light with Light* (Academic Press, NY, 1985).
17. P. Günter and J.-P. Huignard, eds., *Photorefractive Materials and Their Applications* (Springer-Verlag, NY, 1989).
18. A D McAulay, "Real-Time Optical Expert Systems," Appl. Opt. 26, 1927 (1987).
19. G. Eichmann and H. J. Caulfield, "Optical Learning (Inference) Machines," Appl. Opt. 24, 2051 (1985).
20. H. H. Szu and H. J. Caulfield, "Optical Expert Systems," Appl. Opt. 26, 1943 (1987).
21. M W Haney, R A Athale, and R. A. Geesey, "Optical Techniques for Increasing the Efficiency of Heuristic Search," Opt. Eng. 28, 417 (1989).
22. D. Casasent and A. J. Lee, "Optical Relational-Graph Rule-Based Processor for Structural-Attribute Knowledge Bases," Appl. Opt. 25, 3065 (1986).
23. D Casasent and E C. Botha, "Knowledge in Optical Symbolic Pattern Recognition Processors," Opt. Eng. 26, 34 (1987).
24. D. Casasent and S. A. Liebowitz, "Model-Based Knowledge-Based Optical Processors," Appl. Opt. 26, 1935 (1987).
25. D. Casasent and A. Mahalanobis, "Rule-Based Symbolic Processor for Object Recognition," Appl. Opt. 26, 4795 (1987).
26. E Botha, D Casasent, and E. Barnard, "Optical Production Systems using Neural Networks and Symbolic Substitution," Appl. Opt. 27, 5185 (1988).
27. R. A. Schmidt and W. T. Cathey, "Optical Implementations of Mathematical Resolution," Appl. Opt. 26, 1852 (1987).
28. R. A. Schmidt and W. T. Cathey, "Optical Representations for Symbolic Logic," Opt. Eng. 26, 475 (1988).
29. R. A. Schmidt, "System Architecture of an Optical Reasoning System," Opt. Eng. 28, 410 (1989).
30. D P Casasent and E C Botha, "Multifunctional Optical Processor Based on Symbolic Substitution," Opt. Eng. 28, 425 (1989).
31. A Huang, "Parallel Algorithms for Optical Digital Computers," in *Proc , 10th Inter Optical Computing Conf.*, p. 13, IEEE Catalog No. 83CH1880-4 (1983).
32. K.-H. Brenner, "Programmable Optical Processor Based on Symbolic Substitution," Appl. Opt. 27, 1687 (1988).
33. J N Mait and K -H Brenner, "Optical Symbolic Substitution. System Design using Phase-Only Holograms," Appl. Opt. 27, 1692 (1988).
34. T J Cloonan, "Performance Analysis of Optical Symbolic Substitution," Appl. Opt. 27, 1701 (1988).
35. Special issue on neural networks, Appl. Opt. 26, 4909-5111 (1987).
36. J. A. Kottas and C. Warde, "The Infernet: A Neural Network Architecture For Processing Symbolic Information Using Limit Cycle Attractors," submitted to Applied Optics, September, 1989.

10 Publications, Conference Appearances, and Academic Theses Resulting from DARPA/AFOSR Sponsorship

1. M. G. Frey, "A Translation, Rotation, and Scale Invariant Photodetection System," MIT S.B. Thesis, Dept. of Elec. Eng. and Comp. Sci., Cambridge, MA (May 1988).

2. J. A. Kottas and C. Warde, "Trends in Knowledge Base Processing Using Optical Techniques," *Proc., 1989 IEEE Inter. Conf. Sys., Man, & Cyber.* (Cambridge, MA, November 1989), p. 1250.

3. V. E. Shrauger, L. L. Erwin, and C. Warde, "Computer Generated Multiple Phase Level Holograms Using Color Printer Techniques," presented at the Annual Meeting of the Optical Society of America, Boston, MA (November 5–9, 1990).

4. J. A. Kottas, "Limit Cycles in Neural Networks for Information Processing," MIT Ph.D. Thesis, Dept. of Elec. Eng. and Comp. Sci., Cambridge, MA (July 1991).

5. V. E. Shrauger, "Development and Applications of Computer Generated Phase Only Optical Interconnection Elements," MIT S.M. Thesis, Dept. of Elec. Eng. and Comp. Sci., Cambridge, MA (September 1991).

6. J. A. Kottas, "The Spectral Back-Propagation Training Algorithm," submitted to *Neural Computation* (September 1991).

7. J. A. Kottas, "Finite State Machine Based on Limit Cycles in Neural Networks," submitted to *Neural Networks* (September 1991).

8. V. E. Shrauger, L. L. Erwin, and C. Warde, "Computer Generated Multiple Phase Level Holograms Using Color Printer Techniques," submitted to *Applied Optics* (October 1991).

11 List of Personnel

Faculty and Staff

Professor Cardinal Warde

Visiting Scientists

Laura Erwin
Kuang Yi Huang

Post-Doctoral Researchers

Mark Garrett

Grauuate Students

Jenq Yang Chang
R. Scott Hathcock
James A. Kottas
Thomas McNamara
Vernon Schrauger
Doyle Temple

Undergraduate Students

Michael Frey

Support Staff

Margaret Erninian